Análise e Melhoria de Processos de Especificação, Desenvolvimento e Manutenção de Software

Mariana Romba Rodrigues da Costa

Dissertação de Mestrado

Orientador na FEUP: Prof. Américo Azevedo

Mestrado Integrado em Engenharia Industrial e Gestão

2018-07-02

Aos meus pais,

Resumo

A indústria de desenvolvimento de *software* é uma indústria extremamente complexa e relativamente recente, existindo por isso ainda um longo caminho a ser feito, de forma a atingir-se a eficiência que a indústria tradicional já consegue atingir. O presente projeto teve como principal objetivo a análise e melhoria dos processos de Especificação, Desenvolvimento e Manutenção de *software*, de modo a garantir a maximização da proposta de valor da empresa.

O projeto foi realizado numa *start up* que se encontrava a desenvolver uma nova versão da sua plataforma digital, enquanto mantinha a atual, sendo por este motivo um caso de estudo adequado. Uma vez que a empresa estava numa fase *early stage*, a maturidade do seu produto tecnológico ainda não era devidamente robusta. Assim, a empresa necessitava de desenvolver uma versão melhorada da sua primeira plataforma de forma rápida, visto ser peça central e fulcral para o bom funcionamento do negócio. Por estes motivos tornou-se essencial estudar os 3 processos envolvidos na criação de *software*: Especificação, Desenvolvimento e Manutenção, de forma a identificar o estado atual de cada um deles e assim se perceber que ações de melhoria realizar. Decidiu-se observar estes 3 processos em vez de apenas um, pois permite ter uma visão mais holística de todo o processo associado ao desenvolvimento de *software*.

A primeira parte do projeto passou pela identificação dos principais problemas e respetivas causas em cada uma das três fases mencionadas. Para isso, foi necessária a introdução de algumas ferramentas e procedimentos para recolha de dados e apenas posteriormente foi possível a identificação dos problemas. De seguida, e sempre com base na aplicação de ferramentas e conceitos *Lean*, identificaram-se e desenvolveram-se ações de melhoria, sendo que mais de 90% das mesmas foram implementadas. A maioria destas ações passou pela uniformização de processos, criação de *frameworks*, introdução de ferramentas de gestão visual e alteração de processos.

A implementação destas, no período de desenvolvimento da dissertação, retornou resultados extremamente satisfatórios, para o curto tempo do projeto. No caso do processo de especificação, espera-se a obtenção de resultados a longo prazo, dado a natureza do mesmo. No processo de desenvolvimento foi possível, em média, aumentar a taxa de execução dos *sprints* em 17% e reduzir a duração das *daily meetings* em 62%, o que representa melhorias importantes. No que toca ao processo de manutenção, o tempo médio para resolução de erros foi reduzido em mais de 70% o que, em conjunto com as inúmeras restruturações feitas, permitiu o aumento da satisfação das equipas de uma maioria nada ou pouco satisfeita para muito satisfeita.

Espera-se que no futuro este projeto se converta na entrega de funcionalidades de forma rápida, livre de *bugs*, de acordo com as necessidades dos clientes, internos e externos, sendo isto apenas possível com a criação de estruturas robustas por trás de cada um dos diferentes processos. Esta estruturas, criadas agora com o projeto têm assim como objetivo permitir no futuro a identificação rápida de falhas para que se melhorem continuamente os processos e que isto se converta num produto de valor para o cliente.

Analyses and Improvement of *software* Specification, Development and Maintenance processes

Abstract

The software development industry is an extremely complex e relatively recent industry, so there is still a long way to go to achieve the efficiency that the traditional industry can already achieve. The main goal of this project was to analyse e improve software specification, development e maintenance processes, in order to ensure the maximization of the company's value proposition.

The project was carried out in a start-up that was developing a new version of its digital platform, while maintaining the current one, therefore constituting an appropriate case study. Since the company was in an early stage of development, the maturity of its technological product was not yet sufficiently robust. Thus, the company needed to develop an improved version of its first platform quickly, since it is a central piece to the smooth operation of the business. For these reasons it became of paramount importance to study the three processes involved in creating software: Specification, Development e Maintenance, to identify the current state of each of them e thus to perceive what improvement actions to perform. It was decided to observe these 3 processes instead of just one, since it allows a more holistic view of the entire process associated with software development.

The first part of the project went through the identification of the main problems and their causes in each of the three phases mentioned. For this, it was necessary to introduce some tools and procedures for data collection and only then was it possible to identify the problems. Next, and always based on the application of Lean tools and concepts, improvement actions were identified and developed, more than 90% of which were implemented. Most of these actions went through the standardization of processes, creation of frameworks, introduction of tools of visual management e alteration of processes.

The implementation of these, in the period during which the dissertation was developed, returned extremely satisfactory results, for the short time of the project. In the case of the specification process, long-term results are expected given its nature. In the development process it was possible, on average, to increase the execution rate of sprints by 17% and the duration of daily meetings by 62%, which represents important improvements. Regarding the maintenance process, the average time for error resolution was reduced by more than 70%, which, together with the numerous restructurings made, allowed to increase the satisfaction of the teams from an unsatisfactory majority to a very satisfied one.

It is hoped that, in the future, this project will translate into an expedite and bug-free feature delivery, according to the needs of the customers (internal and external). This being only possible with the creation of robust structures behind each of the different processes. These structures, now created with the project, are intended to allow for the rapid identification of failures in the future, so that processes can be continuously improved and converted into a product of value for the customer.

Agradecimentos

Ao Professor Doutor Américo Azevedo, orientador do projeto por parte da Faculdade, por toda a sua disponibilidade, suporte e acompanhamento ao longo do desenvolvimento do projeto, bem como por todas as sugestões fornecidas.

Ao Engenheiro Tiago Craveiro, orientador do projeto na empresa, por toda a confiança, incentivo, suporte e dedicação ao projeto permitindo me implementar as soluções desenhadase mostrando-se sempre disponível para ajudar.

À equipa de Produto, em especial ao Engenheiro Jorge Ferreira, por todo o apoio, ajuda e suporte disponibilizados no decorrer do projeto.

À equipa de desenvolvimento, *account Managers* e restantes colaboradores da HUUB pela colaboração no trabalho desenvolvido.

À HUUB pela oportunidade dada, por tornar os dias de trabalho sempre cheios de boas surpresas, como eventos de equipa e eventos com toda a equipa, pelo bom ambiente e boa disposição de todos.

À minha família por todo o apoio e carinho que me deram sempre, principalmente durante os momentos mais difíceis, com um agradecimento especial ao meu pai por estar sempre disponível, à minha mãe por me apoiar sempre, em todas as minhas decisões, e se mostrar disponível para me ajudar em qualquer momento de necessidade e à minha irmã pelo apoio dado desde sempre.

Aos meus amigos e colegas de Faculdade, por todo o apoio e entre ajuda ao longo dos últimos 5 anos, nunca me deixando desistir.

Aos meus amigos mais próximos, mais recentes ou mais antigos, um especial obrigada por todo o apoio e companheirismo que me deram ao longo do meu percurso académico e pessoal.

Por fim à FEUP e ao corpo docente por proporcionarem um Mestrado Integrado tão completo e com bases tão sólidas que permitiram a realização da presente Dissertação.

Índice de Conteúdos

1 Introdução 1
1.1 Contextualização do mercado 1
1.2 Importância do uso de plataforma digitais em Logística 2
1.3 Objetivos do projeto 3
1.4 Metodologia de desenvolvimento do projeto 4
1.5 Estrutura da dissertação 5

2 Estado de arte 4
2.1 Metodologias de desenvolvimento usadas em projetos de *software* 6
2.1.1 Metodologias Tradicionais 7
2.1.2 Metodologias *Agile* 8
2.2 Conceitos e Princípios *Lean* aplicados a desenvolvimento de Software 12
2.2.1 Princípios *Lean* – aplicação a desenvolvimento de software 12
2.2.2 Ferramentas, técnicas e métodos aplicáveis 16
2.3 Estratégias para integração de qualidade no produto através da eliminação de defeitos e assegurando a qualidade 17
2.3.1 Como fazer a gestão de defeitos em desenvolvimento de *software - Helpdesk* 17
2.3.2 Necessidade e ações a tomar para reduzir o número de erros reportados 19
2.4 Plataformas de apoio à gestão de projetos 20

3 Descrição e Caracterização do Caso de Estudo 22
3.1 Enquadramento 22
3.2 O Projeto 23
3.3 Caracterização “AS IS” 24
3.3.1 Visão geral sob os Processos envolventes na criação de *software* 24
3.3.2 Processo de Especificação de *Software* 24
3.3.3 Processo de Desenvolvimento de Software 25
3.3.4 Processo de Manutenção de *Software* - Atividades corretivas 26

4 Identificação de Problemas 29
4.1 Problemas no Processo de Especificação 29
4.2 Problemas no Processo de Desenvolvimento 30
4.3 Problemas no Processo de Manutenção 33

5 Identificação e Desenho de soluções 37
5.1 Processo de Especificação 37
5.2 Processo de Desenvolvimento 38
5.3 Processo de Manutenção 40

6 Resultados das soluções implementadas 45
6.1 Processo de Especificação 46
6.2 Processo de Desenvolvimento 46
6.3 Processo de Manutenção 48
6.4 Trabalhos realizados consequentes da análise dos resultados 51

7 Conclusões e perspetivas de trabalho futuro 52
7.1 Principais conclusões 52
7.2 Oportunidades de desenvolvimento futuras 54

Referências 55

ANEXO A:Vantagens e desvantagens de plataformas para gestão de desenvolvimentos e *bugs*. 58

ANEXO B: *Flowchart* do funcionamento inicial do processo de *helpdesk* 60

ANEXO C:Inquérito de avaliação do serviço de helpdesk 61

ANEXO D:*Framework* para especificação de requisitos 65

ANEXO E: *Framework* dos cartões para gestão de *sprint* 67

ANEXO F: Estrutura da *framework* para cálculo das taxas de Execução 68
ANEXO G: Estrutura do *Trello* para reportamento de problemas..... 69
ANEXO H: *Framework* para criação de *demos* 70
ANEXO I: *Framework* para realização de testes para assegurar a qualidade das funcionalidades 71
ANEXO J: Exemplo de uma *framework* para reportamento de um problema 72
ANEXO L:Gestão visual de métricas para *helpdesk*..... 73
ANEXO M: Fluxo simples do novo sistema de *helpdesk* 74

Índice de Figuras

Figura 1 - Metodologia seguida para desenvolvimento do projeto 5
Figura 2 - Método Cascata adaptado de (Royce 1970) 7
Figura 3 - Utilização de metodologias de desenvolvimento Agile. 8
Figura 4 - Método *Scrum* 9
Figura 5 - Posicionamento da Huub na cadeia de abastecimento 22
Figura 6 – Processos de desenvolvimento de *software* 24
Figura 7 - Matriz RACI dos 3 processos envolvidos no desenvolvimento de Software, "AS-IS". 28
Figura 8 - Evolução da taxa de execução 31
Figura 9 - Evolução do backlog ao longo dos sprints 32
Figura 10 - Evolução do tempo médio até resolução de problemas reportados via *helpdesk* 34
Figura 11 - Taxa de classificação e satisfação do sistema 36
Figura 12 - Primeira versão do quadro das daily meetings 39
Figura 13 - Matriz RACI dos 3 processos envolvidos no desenvolvimento de Software, "TO BE". 44
Figura 14 - Análise RICE para priorização da implementação das ações de acordo com os processos mais urgentes. 45
Figura 15 - Taxa de execução entre os sprints 7 e 11, para sprints de 2 semanas. 47
Figura 16 - Taxa de execução entre os sprints 11 e 15, para sprints de 1 semana 47
Figura 17 - Duração das daily (7-11 sprints de 2 semanas, 12-15 sprints de 1 semana) 48
Figura 18 - Comparação entre o número de erros resolvidos e o número de erros reportados, em cada sprint 49
Figura 19 - Distribuição do reportamento de erros por equipas 50
Figura 20 - Redução do tempo até resolução de problemas (esquerda) evolução do nível de serviço (SLA) calulado com base num tempo de resolução ideal de até 2 dias (direita). 51

Índice de Tabelas

Tabela 1 - Principais diferenças entre metodologias tradicionais e *Agile* (Boehm 2004; Vijayasarathy e Butler 2016; Royce 1970; Shaydulin e Sybrandt 2017) 6

Tabela 2 - Principais diferenças entre os métodos *Scrum* e *Kanban* (Lei et al. 2015; Ahmad, Markkula, e Oivo 2013; Matharu et al. 2015, Wysk 2018) 11

Tabela 3 - Comparação dos desperdícios identificados na indústria com os identificados em desenvolvimento de *software* (Poppendieck e Poppendieck 2003, 2007; Hibbs, Jewett, e Sullivan 2009). 13

Tabela 4 - Plataformas de comunicação, respetivos caso de uso e vantagens e desvantagens de utilização (Prioridade B -Baixa, M -Média, E – elevada) 27

Tabela 5 - Categorização de cada tipo de erro 34

Tabela 6 - Revisão de Problemas, Causas e Ações (implementadas e não implementadas) 45

Tabela 7 - Resultados obtidos da avaliação dos problemas entre o sprint 1ao 11 e análise do tempo despendido por tipo de problema (em minutos) 49

1 Introdução

O presente projeto de dissertação desenvolveu-se no âmbito da unidade curricular Dissertação do Mestrado Integrado em Engenharia de Gestão Industrial, da Faculdade de Engenharia do Porto. O trabalho decorreu ao longo do segundo semestre do ano letivo de 2017/2018 e foi realizado em ambiente empresarial, numa empresa *startup* a atuar no domínio das plataformas logísticas inovadoras (*startup* HUUB).

Neste capítulo é feito a apresentação e contextualização do projeto, são apresentados os objetivos e a metodologia adotada e por fim apresenta-se a estrutura considerada para o documento.

1.1 Contextualização do mercado

Ao longo dos anos o mercado mundial tem vindo a sofrer acentuadas alterações devido, por exemplo, ao aumento da oferta de produtos e serviços, acesso massificado aos mesmos, globalização dos mercados, grandes desenvolvimentos tecnológicos. O que antigamente significava ter um negócio estável, atualmente já não o significa. Surge então a necessidade de as empresas encontrarem estratégias para entrarem no mercado e/ou se manterem no ativo, isto é, criar vantagens competitivas. Esta vantagem poderá advir, por exemplo, da entrega ao cliente de um produto que o entusiasme (Gautam e Singh 2008), através da reinvenção de modelos de negócio.

Grandes *players* da indústria (empresas maduras e estáveis) têm vindo a perder mercado para as s*tartups* que começam a surgir cada vez mais e a conseguir oferecer serviços de valor acrescentado através da inovação e uso de tecnologias disruptivas. Significa isto que no mundo das empresas procura-se, neste momento, investir fortemente na oferta de serviços e produtos de valor acrescentado, com especial foco nos assentes em tecnologias. Fala-se, mais concretamente, no uso e criação de plataformas online capazes de responder aos mais diversos problemas, não só através de tecnologias inovadoras, mas essencialmente através de conceitos inovadores. Paralelamente, as empresas investem na introdução e aplicação de conceitos, na sua cultura interna e operações diárias, como *Lean*, *Kaizen*, TQM, entre outros, para conseguirem alcançar os objetivos mencionados de uma forma rápida e sustentável.

Assim, neste contexto as empresas procuram, cada vez mais, destacar-se através de soluções com base no uso de plataformas digitais, no formato de aplicações/sites para os clientes conseguirem de forma fácil tirar o melhor proveito do que as empresas têm para oferecer. Quer sejam aplicações "banais" (obtenção de descontos em cadeias alimentares *MacDonald* e *Burguer King*) como para fazerem compras sem que necessitem de sair de casa (*Zara, Farfetch* …), até plataformas elaboradas como a Intercom que permitem captar clientes, isto é, para as empresas conseguirem oferecer um serviço/produto que se adapte às necessidades do consumidor atual, proporcionando-lhe experiências únicas e procurando satisfazer as suas necessidades mais complexas da forma mais simples possível. Este fenómeno surge, em suma, como forma de as empresas oferecem uma quantidade de serviços mais vasta e ainda para conseguirem alcançar um maior número de potencias clientes sem que haja um maior gasto de

recursos, uma vez que substituem funções operacionais por tecnologia e conseguem simular a ideia de exclusividade para cada cliente.

1.2 Importância do uso de plataforma digitais em Logística

A indústria tradicional, isto é de produtos físicos, depende de um vasto conjunto de processos sendo os mais relevantes os logísticos, marketing e vendas e os financeiros. Para que esta tenha sucesso e consiga satisfazer os clientes, cada vez mais exigentes, poderá ser importante alinhar o seu valor ao valor acrescentado oferecido pelas plataformas digitais, plataformas essas que oferecem suporte a vendas, apoio ao cliente, serviços de logística ou ainda uma plataforma que integre o maior número de mais valias num só local. Importa notar que sem serviços de logística os produtos não chegam até ao cliente final sendo, por isso, um dos processos mais importantes na indústria. Assim, para que se entenda um pouco melhor a necessidade do presente projeto será adequado compreender primeiro o que é a logística.

A logística pode ser definida como a gestão de recursos desde que estes são adquiridos, guardados e transportados até ao utilizador final. Os principais intervenientes envolvidos neste processo são os fornecedores, transportadores, armazéns de abastecimento e todos os responsáveis pelo processamento do produto até que este chegue ao consumidor. A logística pode ser ainda entendida como ter a quantidade de bens correta no tempo certo, na localização apropriada para entregá-los ao cliente final. O objetivo é gerir o aprovisionamento de cada cliente de um modo rápido e eficiente de uma parte da cadeia de valor para outra.

A abrangência inerente à logística tem evoluído ao longos dos tempos tendo sido transformada desde os anos 60, com o aumento da complexidade das cadeias de valor e a expansão dos negócios a nível global. O *boom* da tecnologia e a crescente complexidade dos serviços de logística geraram a capacidade e a necessidade de criar *software* para gerir o processo de logística de cada empresa que atua nesta área. Para que se processe um serviço de logística é necessário ligar várias entidades, o fornecedor da matéria, o criador do produto, o retalhista (ou semelhante) e o cliente final. A maioria das empresas nesta área de negócio têm uma oferta para o cliente muito semelhante – transportar produtos de um destino para outro, permitindo o rastreamento dos mesmos. Para que isto aconteça é necessário produzir/ter uma adequada plataforma de *software*. Como a proposta de valor destas empresas é semelhante, sendo pequenas as diferenças, é necessário encontrar uma forma de se conseguirem destacar num mercado já denso e com muita oferta. Existem inúmeras formas de o fazer, tendo algumas empresas destacando-se por isso mesmo.

Considerando que a maioria dos negócios e indústrias dependem de logística, não deveriam as empresas do sector investir significativamente em novas propostas de valor assentes no uso de tecnologias? Não será um acréscimo à proposta de valor se as empresas do sector logístico oferecessem uma plataforma única para uma gestão integrada?

Tome-se como exemplo as necessidades da indústria da moda que é uma indústria bastante competitiva e difícil de entrar, necessitando de serviços de logística extremamente complexos e que ofereçam a possibilidade de transportar as peças de roupa do local de produção para o armazém e, posteriormente, para o cliente final, no menor tempo possível. Nesta indústria o tempo é uma variável chave, pois um pequeno atraso poderá levar a que as roupas se tornem obsoletas, fora de estação e fora de tendência. Para conseguir cumprir requisitos tão exigentes é necessário ter um grande controlo sobre a cadeia de abastecimento, sobre os processos logísticos, sendo para isto é de suma importância ter soluções tecnológicas extremamente completas, livre de *bugs*, fáceis de usar e sempre atualizadas.

O presente projeto foi desenvolvido numa empresa no ramo da logística que serve a indústria da moda e que oferece aos seus clientes uma gestão completa, ponta a ponta, da cadeia de abastecimento de cada um deles, através do uso de uma plataforma digital. Plataforma esta que

se destaca, não por recorrer a uma tecnologia nova, mais sim pela reinvenção de todo um modelo de negócio. Para que a empresa em questão se diferencie, necessita de garantir uma proposta de valor assente numa plataforma digital capaz de satisfazer todas as necessidades de gestão da cadeia logística dos seus clientes. No caso da indústria da moda, gerir a complexidade da cadeia logística com eficiência é particularmente desafiante. Assim, a proposta de valor está assente na capacidade de mascarar para o cliente a complexidade inerente à operação logística e nomeadamente a todos os serviços que se lhe estão associados. Aqui percebe-se que um desenvolvimento de *software* de acordo com as necessidades do cliente, necessidades estas de que estes, em alguns casos, nem se tinham apercebido, torna-se fulcral e é fundamental, nos dias que correm, para o sucesso de qualquer empresa.

Em suma, atualmente para que uma empresa de logística vingue no mercado ter um bom *software* é um dos recursos fundamentais para ter sucesso neste meio e conseguir oferecer propostas inovadoras e de valor acrescentado, uma vez que a maioria das indústrias dependem deste serviço. Assim, é fundamental investir no desenvolvimento de *software* interno e, para que este investimento tenha valor e de forma a otimizá-lo, é necessário planear e gerir muito bem este processo bem como eliminar todos os obstáculos que o impeçam. É neste contexto que o projeto proposto ganha valor, uma vez que pretende apresentar um conjunto de medidas que permitam apoiar o desenvolvimento tecnológico nesta área.

1.3 Objetivos do projeto

Os objetivos do presente projeto procuram alinhar-se com os objetivos internos da empresa utilizada como caso de estudo e na qual o projeto foi desenvolvido, conjuntamente com as necessidades que o próprio mercado exige.

Como referido anteriormente, surge a necessidade de cada vez mais as empresas apostarem no uso de plataformas *online* e, quando estas já existem, é necessário continuamente melhorá-las ou se a plataforma já não servir o propósito corretamente, desenvolver novas.

Quer uma empresa esteja a criar uma plataforma de raiz, enquanto usa uma versão anterior da mesma, quer a produzir novas funcionalidades para uma já existente, é de suma importância gerir a manutenção da mesma apesar de o foco da empresa estar em novos desenvolvimentos.

É pois, neste contexto, particularmente importante conhecer os fatores críticos de sucesso que permitam a uma empresa que desenvolva *software*, ser mais eficiente no lançamento de novas funcionalidades para o mercado, de modo a ser mais competitiva com os seus pares, de acordo com a sua realidade interna. Como tal, pretende-se responder à pergunta de partida "Como melhorar os processos de especificação, desenvolvimento e manutenção do *software* da empresa?" sendo o objetivo geral deste projeto o de identificar problemas, analisar e aplicar soluções para melhorar os processos de especificação, desenvolvimento e manutenção de *software*. Este objetivo principal pode ser desagregado nos objetivos parcelares seguintes:

- (O1) Identificar os problemas inerentes às fases de especificação, desenvolvimento e manutenção de desenvolvimento de *software*;
- (O2) Identificar e aplicar soluções para os problemas encontrados nos processos de especificação, desenvolvimento e manutenção de desenvolvimento de *software*;
- (O3) Verificar a existência de melhorias após a aplicação das soluções identificadas para os processos de especificação, desenvolvimento e manutenção de desenvolvimento de *software*.

A exploração deste conjunto de objetivos conduzem à análise ao âmbito de três grandes processos. Com a análise do primeiro processo pretende-se construir um modelo normalizado para o processo de especificação de requisitos, de forma a que este garanta que todos os objetivos e diretrizes de desenvolvimento estejam bem definidas e que as necessidades dos

stakeholders do produto a desenvolver/ novas funcionalidades sejam garantidas da melhor maneira.

A análise do segundo processo tem como objetivo a identificação e eliminação/alteração de processos que possam impactar direta ou indiretamente no desenvolvimento de *software*, pela introdução e implementação de ferramentas de gestão visual, entre outras, bem como a medição do seu impacto para o desenvolvimento de *software*, de modo a acelerar o seu processo.

Por fim, o terceiro processo analisado é a gestão do modelo de suporte à plataforma que vigora no momento. Com o objetivo de criação de um serviço de apoio ao cliente (*helpdesk*), que permita não só mitigar o impacto desta manutenção na equipa de desenvolvimento (a contruir a nova plataforma), mas também sem deixar que isto comprometa o correto uso da atual plataforma, para que esta não se desalinhe com as necessidades dos seus utilizadores.

A base de todo este projeto estará assente no estudo e aplicação de princípios e conceitos já estudado, nomeadamente *Lean* e todos os conceitos que o envolvem.

1.4 Metodologia de desenvolvimento do projeto

O presente projeto foi pensado, desde o seu início, numa lógica de identificação de problemas e resolução dos mesmos, dividindo-se o processo em várias fases.

Numa primeira fase começou-se pela observação dos processos, no terreno, associados ao desenvolvimento de *software* de modo a se compreender a realidade operatória. Paralelamente a este processo, iniciou-se uma análise à literatura relevante e à recolha de alguns dados importantes para o diagnóstico dos processos.

Numa segunda fase confrontaram-se os dados obtidos e os fluxos do processo com o conhecimento obtido através da literatura. Utilizaram-se assim conceitos chave que permitiram a identificação de problemas, o que resultou numa abordagem dedutiva, aplicando-se conceitos e teorias já desenvolvidas e estudas pelo vasto conjunto de autores e aplicando-os ao caso de estudo em questão. Nesta fase procedeu-se ao desenvolvimento de *workshops* e *surveys* para complementar a identificação dos problemas.

Numa terceira fase e mais uma vez com o apoio da literatura, desenharam-se um conjunto de soluções com o objetivo de se eliminarem estes problemas, alinhando-se isto aos objetivos da empresa em estudo.

Na fase seguinte implementam-se as soluções e recolhem-se os seus resultados validando-se a efetividade das medidas tomadas e em alguns casos verificando determinadas hipóteses.

Posteriormente através dos resultados obtidos foi possível propor e, em alguns casos, implementar novas soluções para melhoria de alguns novos problemas detetados após a obtenção de resultados das medidas já aplicadas.

A Figura 1 ilustra o processo seguido.

Figura 1 - Metodologia seguida para desenvolvimento do projeto

1.5 Estrutura da dissertação

O presente capítulo pretende contextualizar o mercado atual focando-se na forte competitividade do mesmo e na importância do desenvolvimento de boas plataformas digitais para que as empresas vinguem no seu meio, justificando assim a necessidade e urgência da remoção de desperdícios e da introdução de práticas que apoiem o rápido desenvolvimento de *software*.

O capítulo seguinte tem como objetivo apresentar o conceito e princípios *Lean*, já aplicados em desenvolvimento de *software*, bem como o estudo de metodologias de desenvolvimento que, de acordo com o tipo de projeto em mão poderão ser mais ou menos adequadas, servindo de apoio à tomada de decisão da(s) metodologia(s) adequada para o caso de estudo em causa. Neste capítulo estudou-se ainda um vasto conjunto de ferramentas que poderão ser aplicadas bem como medidas preventivas e corretivas para uma plataforma sem falhas.

De seguida, no terceiro capítulo, é apresentada a empresa utilizada como caso de estudo, bem como a explicação detalhada sobre a sua forma de trabalho no que diz respeito ao desenvolvimento de *software*.

O capítulo subsequente, o quarto, foca-se na identificação de oportunidades de melhoria, a ser aplicadas nas diversas etapas do processo de desenvolvimento ou em processos que impactem sobre o mesmo.

No capítulo cinco são identificadas as ações aconselhadas a tomar como forma de mitigar ou eliminar os impactos negativos dos problemas.

O capítulo sexto surge para apresentar os resultados obtidos com a implementação das ações identificadas no capítulo cinco, bem como novas soluções para melhoria continua dos processos.

Por fim, o sétimo capítulo tem por objetivo a reflexão e a retirada de conclusões obtidas bem como a perspetiva de trabalhos futuros.

2 Estado de arte

O presente capítulo tem como objetivo primeiramente explorar, de forma sucinta, as metodologias de desenvolvimento utilizadas em *software*, fornecendo uma visão global de quais os métodos mais utilizados atualmente bem como os casos de utilização e vantagens e desvantagem de cada um deles.

De seguida é feita uma abordagem aos conceitos *Lean* apresentando a adaptação dos mesmos, da indústria para a aplicação no desenvolvimento de *software*, paralelamente a algumas práticas e ferramentas que se podem adotar para alcançar os objetivos proposto pela aplicação destes conceitos.

Por fim é feita uma revisão de ferramentas tecnológicas que podem ser utilizadas para apoio à gestão de projetos, com foco nos projetos de desenvolvimento de *software.*

2.1 Metodologias de desenvolvimento usadas em projetos de *software*

De uma forma geral existem dois grandes grupos de metodologias de desenvolvimento, as pesadas (tradicionais) e as leves (*agile*) (Shaydulin e Sybrandt 2017). Dentro de cada uma delas existe um vasto número de métodos que se regem pelos mesmos princípios ou por princípios semelhantes. Assim pode dizer-se que as grandes diferenças entre estes dois tipos de metodologias residem na velocidade da entrega de *features*, no tamanho das equipas necessárias para a execução de projetos, no orçamento necessário e na sequência dos passos a seguir desde o início do projeto até à sua conclusão (Vijayasarathy e Butler 2016).

Na Tabela 1 é feita uma comparação direta de acordo com os pontos que mais diferenciam os dois tipos de metodologias.

Tabela 1 - Principais diferenças entre metodologias tradicionais e *Agile* (Boehm 2004; Vijayasarathy e Butler 2016; Royce 1970; Shaydulin e Sybrandt 2017)

Fator	Metodologias Tradicionais	Metodologias Agile
Tamanho da Organização	Elevado número de funcionários Receitas elevadas	Baixo número de funcionários Receitas moderadas
Orçamento	Elevado	Baixo
Equipa:	Muitos elementos – diversas equipas.	Poucos elementos – 1 equipa
Tamanho **Outras características**	Elevado nível de formalidade (ex.: muita documentação) Funções de cada elemento da equipa bem definidos. Não é necessário uma equipa com elevado nível de experiência (Boehm 2004)	Elevado nível de experiência e autonomia, quer a nível técnico como processual, o que permite a liberação de algumas formalidades como alguns tipos de documentação (Boehm 2004)
Velocidade de entrega	Entrega de um produto final – o produto só é lançado para o mercado depois de estar completo. Isto leva a que o produto demore muito tempo a chegar ao mercado o que poderá trazer consequências como torná-lo obsoleto.	Entregas constantes - há medida que as *features* vão estando prontas, fazem-se *releases* do produto, o que permite um lançamento bastante mais rápido para o mercado.

É importante perceber que ambas as metodologias têm os seus pontos fortes e fracos, e que a Tabela 1 apresenta apenas as características de dois extremos opostos, não tendo em conta a existência de proximidade de algumas metodologias tradicionais com as *agile* e vice-versa.

Outro ponto importante a ter em conta são as características que devem ser analisadas na escolha da metodologia de desenvolvimento a utilizar (Vijayasarathy e Butler 2016). São elas:

- A Organização;
- A equipa de desenvolvimento;
- As necessidades do projeto.

Para além disto, é necessário ter em atenção que, por vezes, a solução reside no uso híbrido destas metodologias, não sendo obrigatório nem necessário cumpri-las na integra (VersionOne, 2018).

2.1.1 Metodologias Tradicionais

Waterfall

O Desenvolvimento *Waterfall* ou como é conhecido em português, Desenvolvimento em Cascata, é um método tradicional utilizado para gerir desenvolvimentos de produtos (Royce 1970). Em 1970, Royce transportou da indústria este método, descrevendo a sua aplicação para o desenvolvimento de *software*.

O método *Waterfall* é composto por 7 fases, visíveis na Figura 2, distribuídas numa sequência rígida, isto é, só quando uma fase do projeto é dada como concluída é que se inicia a seguinte. É por este motivo considerado um modelo rígido e pouco flexível (Royce 1970; Shaydulin e Sybrandt 2017).

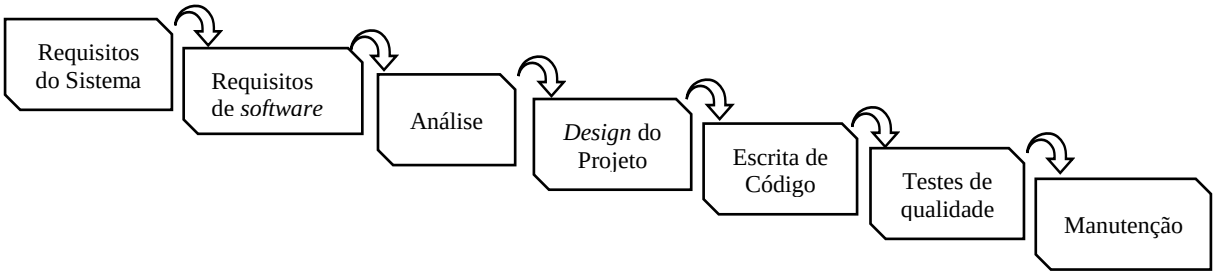


Figura 2 - Método Cascata adaptado de (Royce 1970)

Para este ser o método escolhido a organização em causa deve reunir o seguinte conjunto de características (Vijayasarathy e Butler 2016):

- Gerar elevadas receitas;
- Orçamento elevado;
- Várias equipas de grande dimensão;
- Projeto de elevada criticidade e com possibilidade de o estruturar de uma ponta a outra e executá-lo de forma sequencial (Adel e Abdullah 2015; Vijayasarathy e Butler 2015; Royce 1970; Shaydulin e Sybrandt 2017).

De acordo com diversos autores (Estadual et al. 2012; Royce 1970) a utilização deste método para desenvolvimento de software é incompatível com a aproximação de um desenvolvimento com base em conceitos *Lean* e traz desvantagens como as que se referem a seguir (Adel e Abdullah 2015; Mohammad, Munassar, e Govardhan 2010):

- Não permite iterações entre as diferentes fases;
- Não permite modificações de requisitos;
- Todo o processo é afetado pelo atraso numa fase;
- Baixa visibilidade sobre os problemas – testes só são feitos numa fase final;
- Entrega do projeto no fim – requisitos do cliente podem ter mudado ou ser diferentes do que os levantados.

Apesar das desvantagens enumeradas, este método é ainda muito usado (Shaydulin e Sybrandt 2017). De acordo com o estudo de Vijayasarathy e Butler (2016) este método é utilizado em 32% dos projetos.

2.1.2 Metodologias *Agile*

As metodologias *Agile* surgiam como forma de dar resposta à incapacidade das metodologias tradicionais para o desenvolvimento de software (Hneif e Ow 2009).

Dentro deste tipo de metodologias existe um vasto número de outras que se regem pelos princípios que deram origem à metodologia *Agile* (Hibbs, Jewett, e Sullivan 2009). São estas Scrum, Extreme Programming, Kanban, Agile Modeling, Crystal, Rapid Application Development, Feature Driven Development, Lean Software Development, Test Driven Development e muitos outros (Silva, Schramm, e Damasceno 2016). A somar a estas, surge ainda um vasto conjunto de combinações entre as mesmas.

A Figura 3 apresenta o conjunto de metodologias *agile* mais usadas no mercado hoje em dia, de acordo com o "12th Annual State of Agile Report" (VersionOne, 2018) que revela que 52% das empresas, inquiridas, usa métodos *agile* em pelo menos metade das suas equipas, pelo que, em seguida se explora os métodos Scrum e Kanban, devido à sua importância no mercado referindo as características que os tornam adequados para o presente projeto.

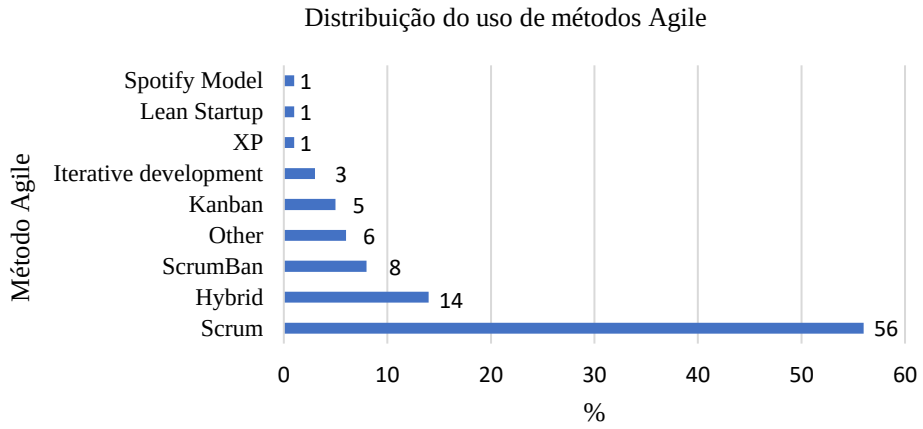


Figura 3 - Utilização de metodologias de desenvolvimento Agile.

Scrum

Nascida nos anos 90, *Scrum* é uma framework composta, por 5 eventos: *Sprint, Sprint planning, Daily Scrum, Sprint Review e Sprint Retrospective* (Schwaber e Sutherland 2013). A Figura 4, de forma sucinta, mapeia a sequência de processos de uma ponta à outra.

Intervenientes

Esta metodologia pressupõe o envolvimento de 3 entidades distintas, são elas a equipa de desenvolvimento, o *Product Owner* e o Scrum Master. Cada uma destas entidades tem um papel distinto e fundamental para o bom desenrolar de todo o processo de desenvolvimento (Schwaber e Sutherland 2013; Matharu et al. 2015).

O *Product Owner* é o responsável pela definição, priorização e comunicação dos requisitos para o(s) produto(s) em desenvolvimento (Matharu et al. 2015).

A equipa de desenvolvimento é composta por 3 a 9 elementos e é responsável pela execução das tarefas que são alocadas para cada *Sprint*, pelo *Product Owner* (Matharu et al. 2015).

O *Scrum Master* é a terceira figura deste enredo assumindo a responsabilidade de "forçar" o cumprimento do que foi planeado para o *sprint* prestando apoio na eliminação de impedimentos ao desenvolvimento, por parte da equipa de desenvolvimento, bem como o apoio ao próprio desenvolvimento das tarefas propostas para o *sprint* (Matharu et al. 2015).

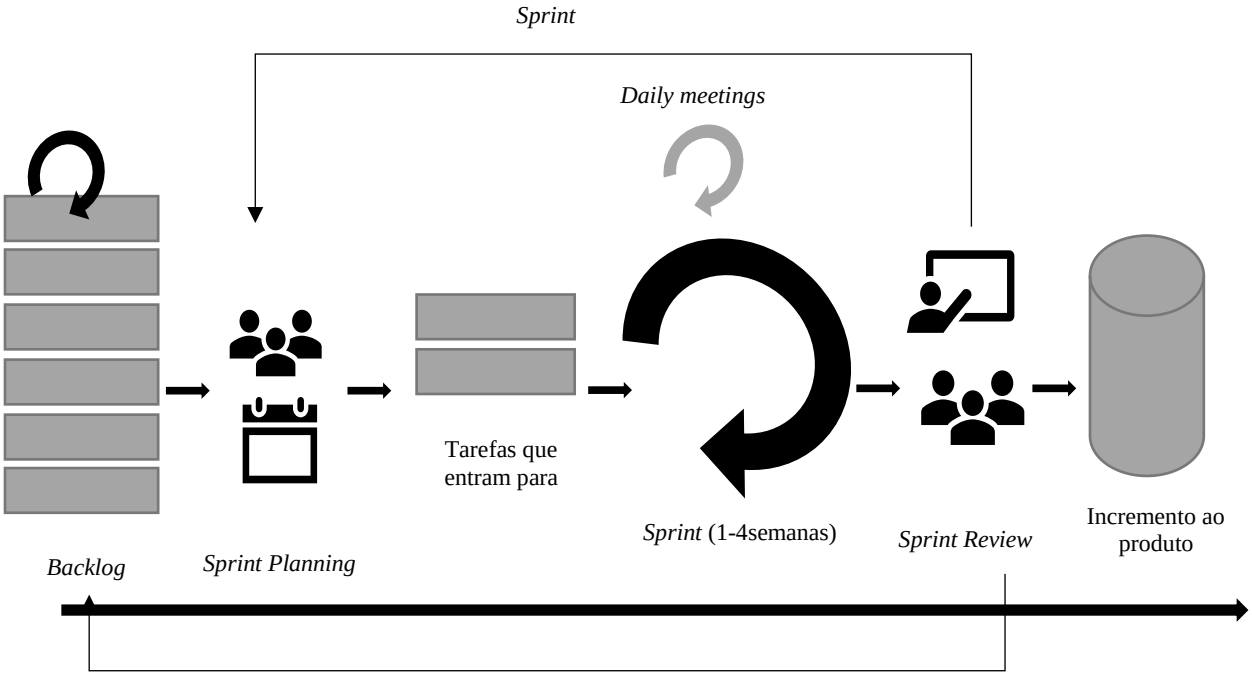


Figura 4 - Método *Scrum*

Sprint

Esta *framework* pressupõe a gestão do desenvolvimento de *software* por iterações - os *sprints*. Cada *sprint* tem a duração máxima de 1 mês (4 semanas), variando a duração de empresa para empresa, mas sendo constante para um mesmo projeto (Schwaber e Sutherland 2013). Cada *sprint* compreende um ciclo completo de produção desde a conceptualização à implementação do produto em causa (Matharu et al. 2015). Durante cada ciclo pressupõe-se (Schwaber e Sutherland 2013) :

- A não alteração de atividades que possam pôr em causa ou comprometer o alcance do *Sprint Goal*;
- Os objetivos de qualidade não devem ser diminuídos;
- O esboço do *sprint* deve poder ser clarificado e redefinido pelo *Product Owner*;
- Criação de conhecimento, por parte da equipa, através da aprendizagem com a resolução das dificuldades com as quais se vão deparando.

Sprint Planning

Para que cada *sprint* possa ocorrer dentro do espectável é necessário planeá-lo – *Sprint Planning*. Este evento deverá ter uma duração máxima de 8h (para o planeamento de um *sprint* de 1 mês, e quanto menor o tempo do sprint menor deverá ser o tempo gasto no seu planeamento). Deste evento deve resultar a resposta a duas questões (Schwaber e Sutherland 2013): "O que pode ser feito no *Sprint*?" e "Como é que o trabalho escolhido vais ser feito?".

Daily meeting

As *daily meeting* são reuniões diárias, que acontecem ao longo do *sprint*, que servem para planear as 24 horas seguintes permitindo alinhar a equipa sobre o estado de desenvolvimento e assim ser alcançando o *Sprint Goal*, atingindo a produtividade desejada. Estas reuniões devem ter uma duração máxima de 15 minutos, com a presença de todos os membros da equipa de desenvolvimento incluindo o *Scrum Master* (moderador da reunião) e o *Product Owner* (Schwaber e Sutherland 2013; Matharu et al. 2015).

A agenda destas reuniões deverá compreender a resposta, de cada um dos membros da equipa de desenvolvimento sobre as seguintes questões:

- O que fiz ontem para ajudar a alcançar o *Sprint Goal*?
- O que vou fazer hoje para ajudar a alcançar o *Sprint Goal*?
- Que impedimentos afetam o meu trabalho de hoje?

Para uma discussão mais detalhada é comum que, após estas reuniões, alguns membros se reúnam para discutir, adaptar ou replanear o resto do trabalho do *Sprint*, de acordo com as informações que obtiveram na *daily meeting* (Schwaber e Sutherland 2013).

As principais vantagens desta prática são o alinhamento de toda a equipa, identificação de impedimentos no desenvolvimento de *software*, promoção de decisão rápida, promoção do aumento do nível de comunicação e ainda eliminação de outras reuniões (Schwaber e Sutherland 2013).

Sprint Review

Os *Sprint Reviews* ocorrem no final de cada iteração, *Sprint*, sendo o principal objetivo destes a revisão do *Sprint* como o próprio nome indica. Este objetivo é cumprido através da colaboração entre equipa de desenvolvimento e *stakeholders* (Schwaber e Sutherland 2013).

Nestes encontros/reunião é comum que (Schwaber e Sutherland 2013):

- O *Product Owner* faça uma revisão do *sprint* apresentando, das tarefas planeadas, as que foram executas e as que não foram;
- A equipa de desenvolvimento exponha e discuta o que correu bem e o que correu mal durante o *Sprint*;
- A equipa de desenvolvimento demonstre o trabalho executado;
- O *Product Owner* discuta o estado do *Product Backlog*;
- Todos os presentes discutem o que deve entrar para o *Sprint* seguinte;
- Revisão de como as mudanças no mercado ou da potencial utilização do produto possam ter mudado e, consequentemente, o alinhamento das prioridades;
- Revisão do horizonte temporal, orçamento, capacidade da equipa e mercado para os lançamentos do produto futuro.

Estes encontros não têm estatuto oficial de reuniões, devendo ter no máximo uma duração de 4 horas, para um *Sprint* com duração de 1 mês. Mais uma vez, e tal como nas *daily meeting*, o *Scrum Master* tem aqui o papel de garantir que as reuniões se realizem e que não ultrapassem o tempo estipulado. Do *Sprint Review* deve sair um *Product Backlog* que defina o trabalho que irá ser realizado no(s) *Sprint(s)* futuro(s) (Schwaber e Sutherland 2013).

Sprint Retrospective

A *Sprint Retrospective* ocorre depois de um *Sprint Review* e antes de um *Sprint Planning*. Os principais objetivos são:

- Rever como correu o *Sprint* em termos de relações interpessoais, processos e ferramentas;
- Identificar e ordenar as coisas que correram melhor bem como potenciais melhorias;
- Criar um plano para melhorar o desempenho do *Scrum Master*.

Tal como nos *Sprint Review* o *Scrum Master* tem o papel de assegurar que o encontro tome lugar e que o faça dentro dos limites temporais previamente previstos. O principal *output* desta reunião deve ser um conjunto de medidas de melhoria para o funcionamento da equipa de *Scrum*. Isto não significa que ao longo do *Sprint* isto possa acontecer, mas este é o momento oficial para acertar todos esses pormenores (Schwaber e Sutherland 2013).

Kanban

Kanban é uma metodologia que tem vindo a ganhar maior expressão no mercado. Esta é também considerada um método *agile* e uma ferramenta *Lean,* regendo-se por 6 princípios (Lei et al. 2015; Matharu et al. 2015):

- Visualizar o trabalho necessário;
- Limitar o trabalho em Progresso (WIP);
- Medir e gerir o fluxo;
- Tornar explícitas as políticas de processo;
- Melhorar a colaboratividade;

Os dois primeiros princípios são sustentados com o uso de ferramentas visuais, o quadro *Kanban,* que permite de uma forma simples visualizar e limitar o trabalho em progresso (WIP), uma vez que esta metodologia pressupõe a concretização das tarefas apenas quando necessário, para reduzir o lixo produzido e limitar o WIP (Matharu et al. 2015).

Tal como qualquer método *agile,* pressupõe o lançamento constante de novas *features*/produtos, contrariamente ao que é feito noutras metodologias que o fazem em *batch* (Matharu et al. 2015).

Este método distingue-se de outros pela não existência de papéis atribuídos aos membros da equipa *tech,* sendo da responsabilidade de toda a equipa o cumprimento dos prazos estabelecidos de entrega dos produtos, havendo grande flexibilidade na alteração do planeamento (Matharu et al. 2015; Ahmad, Markkula, e Oivo 2013).

A Tabela 2 sumaria as principais diferenças entre os métodos *Scrum* e *Kanban*, sendo que uma das maiores diferenças entre eles é o facto de o método *Kanban* permitir mudanças a qualquer momento, sendo considerado adequado para a resolução de bugs/erros (Wysk 2018).

Tabela 2 - Principais diferenças entre os métodos *Scrum* e *Kanban* (Lei et al. 2015; Ahmad, Markkula, e Oivo 2013; Matharu et al. 2015, Wysk 2018)

Parâmetros	***Scrum***	***Kanban***
Fluxo de trabalho	Iterações – *sprints*	Pequenas iterações
Entrega do produto	Entregas a cada *sprint*	Entregas contínuas
Coordenador do projeto	*Scrum Master*	Equipa
Mudanças	Não são permitidas durante o *sprint*	Permitidas a qualquer momento
Casos de uso	Funcionalidade definida (projeto de menor risco)	Incertezas na funcionalidade (projeto de maior risco)

Os dois métodos, *Scrum* e *Kanban*, têm como principal ponto comum o facto de serem caracterizadas como *Agile* e *Lean* simultaneamente (Lei et al. 2015, Wysk 2018).

Técnicas mais usadas no mercado

De acordo com o "12th Annual State of Agile Report" (VersionOne, 2018) as 5 técnicas *Agile* mais usadas no mercado são: Reuniões diárias – 90%; *Sprint /iteration planning* – 88%; *Retrospectives* – 85%; *Sprint iteration review* – 80%; pequenas iterações – 69%. O uso de ferramentas/técnicas Kanban acontece em 65% dos casos.

2.2 Conceitos e Princípios *Lean* aplicados a desenvolvimento de Software

No Japão, no século XX, o conceito *Lean* foi intensamente desenvolvido numa pequena unidade fabril, a *Toyata* (Ohno 1988). Esta fábrica tentava, na altura, produzir carros a um preço baixo, para conseguir ser competitiva mas sem ter de os produzir em massa, como fazia a *Ford* e outras fábricas Americanas, uma vez que o mercado japonês não tinha dimensão suficiente para consumir todos esses carros (Poppendieck e Poppendieck 2003). O enorme sucesso das práticas implementadas nesta fábrica de automóveis levou ao estudo e replicação das mesmas nos mais diversos tipos de indústria, dando origem ao conceito de *Lean Thinking* (Womack e Jones 1990). Este conceito *Lean* só foi difundido mundialmente com a publicação do livro "A Máquina que Mudou o Mundo" (Womack e Jones 1990).

A aplicação deste conceito pressupõe que as organizações se foquem no cliente, para que o consigam entender e satisfazer, de modo a conseguirem identificar quais os processos de valor que devem manter e quais os processos que devem eliminar (Hines et al. 2008). A eliminação dos desperdícios mencionados sugere a redução dos custos bem como a redução do tempo de entrega, tal como foi conseguido na Toyota.

O conceito de *Lean* originalmente é constituído por 5 princípios, criar valor, definir a cadeia de valor, otimizar o fluxo, sistema *pull* e perfeição (Womack e Jones 1996). Mary e Tom Poppendieck (2003, 2007) transpuseram estes princípios para o desenvolvimento de *software*, convertendo-os em 7 princípios. Por este motivo estes 2 autores são considerados os "pais" dos mesmos e são citados por inúmeros outros (Hibbs, Jewett, e Sullivan 2009; Ahmad, Markkula, e Oivo 2013).

2.2.1 Princípios *Lean* – aplicação a desenvolvimento de software

A metodologia *Lean*, aplicada a desenvolvimento de *software*[1], rege-se por 7 princípios, trabalhados de modo a serem traduzidos em práticas *Agile*, sendo comummente usados em conjunto com outros métodos *Agile* (Poppendieck 2012). São eles:

- **Eliminar os desperdícios**:

Este princípio refere-se, de um modo geral, à redução de tudo o que possa representar um desperdício, na "produção" de um produto. Considera-se como um desperdício tudo aquilo que não represente valor para o cliente final, apesar de puder ser algo que a empresa considere como importante ou até mesmo essencial (Ohno 1988).

No caso da indústria, este desperdício está muitas vezes na quantidade excessiva de *stock* e, no caso do *software*, no excesso de funcionalidades sem utilidade direta para o cliente. De acordo com Mary e Tom Poppendieck (2007) apenas 20% das funcionalidades desenvolvidas são

[1] É importante evidenciar que existe alguma discordância quanto à hierarquia de métodos devido às grandes similaridades entre os princípios *Agile* e *Lean*, existindo autores que argumentam que *Agile* pertence a *Lean* e outros que argumentam o contrário (Kupiainen, Mäntylä, and Itkonen 2015).

efetivamente usadas pelos utilizadores. Uma forma de eliminar este problema é através da inclusão do cliente (interno ou externo) para a tomada de decisões relativas ao produto, uma vez que ajuda na priorização dos requisitos (Hibbs, Jewett, e Sullivan 2009).

Mary e Tom Poppendieck (2007) evidenciam 7 grandes causas de desperdício no desenvolvimento de *software*, que se passam a referir: trabalho incompleto, funcionalidades excessivas, processos desnecessários, troca de tarefas, atrasos, defeitos e movimentação, fazendo posteriormente o seu paralelismo com as 7 causas de desperdício na indústria. O estudo de Hibbs, Jewett e Sullivan (2009) vem no mesmo sentido.

Assim, na Tabela 3 apresenta-se um sumário e uma breve comparação entre os desperdícios encontrados na indústria e no desenvolvimento de *software*.

Tabela 3 - Comparação dos desperdícios identificados na indústria com os identificados em desenvolvimento de *software* (Poppendieck e Poppendieck 2003, 2007; Hibbs, Jewett, e Sullivan 2009).

Desperdício Indústria	**Desperdícios Software**	**Problemas**
Inventário	Trabalho incompleto	Acumulação de tarefas em *backlog*, principalmente *backlog* de bugs. Quanto mais rápido um bug for resolvido menor a probabilidade de se fazer desenvolvimentos em cima do mesmo.
Superprodução	Funcionalidades excessivas	Gasto de recursos desnecessários.
Transportação	Troca de tarefas	Perda de concentração e dificuldade em retomar as tarefas
Movimentos	Movimentos	Equipa *tech* estar isolada da restante equipa leva a que para comunicar com a mesma seja necessários movimentos desnecessários.
Tempo de espera	Tempo de espera	Possibilidade do produto se tornar obsoleto, principalmente em *software.*
Defeitos	*Bugs*	Retrabalho.
Processos desnecessários	Desenvolvimentos desnecessários	Gasto de recursos desnecessários.

De um modo geral, a eliminação de todos os processos que não tragam valor para o cliente permite cumprir este primeiro princípio *Lean* – eliminar desperdícios.

- **Integrar qualidade:**

Qualidade, no desenvolvimento de *software*[2], pode ser definida como o cumprimento dos requisitos especificados para o produto em questão e o seu correto funcionamento (Galin 2018).

Para que este princípio seja cumprido é importante entregar ao cliente, um produto sem qualquer tipo de falhas. No caso do *software* estas falhas denominam-se, muitas vezes, de *bugs*. Mais importante que procurar este tipo de defeitos, é preveni-los (Poppendieck e Poppendieck 2007). Regra geral, é mais penoso, quer na indústria quer no desenvolvimento de *software* corrigir os erros do que evitá-los, não só a nível de tempo como de gasto de outros recursos (monetários, humanos, entre outros…).

[2] Existem inúmeras *frameworks* para "medir" qualidade como a ISO 9000, *Capability Maturity Model, Capability Maturity Integration, Model McCall's* e muitos outros.

De acordo com a IBM (Steffen, 2011) algumas das práticas a serem implementadas passam pela verificação de defeitos ao longo do processo e não apenas no final do mesmo, corrigir os defeitos encontrados no momento (o que fica mais barato) e ainda focar-se na prevenção em vez da correção. Algumas das práticas mais concretas, sugeridas por esta entidade, são: *refactoring*, integração continua, *code review* e a realização de um variado conjunto de testes (contínuos e automatizados). Algumas destas recomendações são também feitas por outros autores tal como Mary e Tom Poppendieck (2003) que para além dos atrás mencionados, referem a realização de testes o mais cedo possível e ainda, em vez da recolha de mais requisitos dos utilizadores, mostrar-lhes alguns protótipos de modo a recolher os seus *feedbacks*. Outras medidas são descritas, mais à frente, no subcapítulo 2.2.3.

- **Criar conhecimento:**

Quando se desenvolve uma determinada funcionalidade ou resolve um problema, aprende-se sempre algo com isso, dai que a segunda vez que, por exemplo se resolve um problema semelhante, o tempo gasto é consideravelmente menor. Para que isto seja possível é necessário, de algum modo, armazenar esse conhecimento para que fique disponível para o futuro e assim evitar o desperdício de tempo na reconstrução de uma solução já previamente encontrada/desenvolvida (Poppendieck e Poppendieck 2003).

De acordo com a IBM (Steffen, 2011) este princípio é melhor conseguido através da criação de equipas de desenvolvimento pequenas e *cross-functional*[3], criação de *guidelines/frameworks*, *code review*, compartilha de informação, *mentoring* e ainda ciclos de *feedback* e inspeções. Alguns autores sugerem ainda a recolha intensiva de informação (Hibbs, Jewett, e Sullivan 2009; Poppendieck e Poppendieck 2003) através:

- Do diálogo constante com o cliente, interno e/ou externo, de modo a recolher o seu feedback para que a equipa compreenda a que distância se encontra do objetivo;
- Do diálogo com as diversas equipas envolvidas no produto (equipa de desenvolvimento, de QA, de produto, e outras adequadas) de modo a recolher feedback e avaliar formas de melhoria contínua;
- Da colocação um diagrama, respeitante ao projeto atual, numa área comum, de modo a que toda a equipa o consiga ver;
- Da sugestão de que quando a equipa de desenvolvimento se depara com um problema encontre três possíveis soluções e que as teste.

Algumas metodologias, como as *Agile*, mais especificamente a *Scrum*, já pressupõe grande parte deste tipo de práticas (ver Capítulo 2.1.2).

- **Adiar comprometimentos/decisões:**

O desenvolvimento de *software* pressupõe um enorme conjunto de incertezas. Adiar a tomada de decisões irá permitir sustentá-las em factos em vez de apenas especulações. No desenvolvimento de *software* este princípio é fundamental, uma vez que quanto mais tarde for tomada uma decisão, por exemplo, no desenvolvimento ou não de uma determinada funcionalidade, mais acertada será a mesma. A decisão tardia poderá evitar o retrabalho, devido à possível necessidade de alterar algumas das especificações previamente feitas, ou até mesmo o trabalho em si caso se decida que a funcionalidade desenhada não deva ser produzida, podendo ainda evitar outros tipos de complicações (Poppendieck e Poppendieck 2003; Hibbs, Jewett, e Sullivan 2009).

[3] Equipas *cross-functional* são equipas que possuem um vasto conjunto de competências que se completam umas às outras.

Algumas das práticas sugeridas pela IBM (Steffen, 2011) passam por: desenvolvimento por iterações (ver capítulo 2.1.2), reuniões de planeamento, entre outros.

- **Entrega rápida:**

Este princípio refere-se à entrega do produto, ao cliente, no tempo adequado. Na área do desenvolvimento de *software,* todos os dias surgem centenas de novas aplicações para "resolver" os mais diversos problemas, como tal o *timing* de entrega é essencial para que um produto tenha sucesso.

Para que este princípio seja alcançado é necessário um elevado nível de organização, uma forma de trabalho padronizada e acima de tudo a descrição e especificação do trabalho numa linguagem adequada (Poppendieck e Poppendieck 2003). A IBM (Steffen, 2011) sugere também a utilização de um sistema *Pull* e desenvolvimento por iterações – iterações curtas permitem não só cumprir este princípio como eliminar desperdícios e ainda adiar compromissos (Hibbs, Jewett, e Sullivan 2009) (1º,3º e 5º princípios *Lean).*

Algumas ferramentas que permitem ajudar a compreender a posição da equipa face ao trabalho que tem de ser feito para desenvolver todo o processo são a criação de um local onde qualquer individuo interessado no projeto possa ter acesso ao que está a ser desenvolvido bem como ao objetivo geral do projeto.

A identificação do *bottleneck* do processo é também essencial para gerenciar o tempo e alocação de recursos, para posteriormente evitar atrasos na entrega do projeto.

- **Dar poder à equipa:**

Este principio pressupõe que seja dada "voz" às equipas de modo a que estas possam ativamente participar na tomada de decisões (Poppendieck e Poppendieck 2003). Quer isto dizer que se pretende o envolvimento das diversas partes, na tomada de decisões, fomentando a auto-organização e mostrando confiança na equipa.

A IBM (Steffen, 2011) sugere a criação de um ambiente de respeito e envolvimento das equipas alcançado através da autogestão, trabalho de equipa e a troca de *feedback* entre os elementos de cada equipa.

- **Otimizar o todo:**

À medida que um produto vai sendo desenvolvido não interessa a qualidade individual de cada uma das partes que o compõe, mas sim da qualidade obtida quando as mesmas são integradas. (Poppendieck e Poppendieck 2003).

Para cumprir este objetivo a IBM (Steffen, 2011) sugere o uso de métricas, tempo de ciclo + Mapa de fluxo de Valor; ROI + Modelo de Lucros e Perdas, Satisfação do cliente + entendimento das suas necessidades e a valorização de métricas de desempenho da equipa em vez de individuais.

É importante notar que, tal como o último princípio o enuncia, os princípios *Lean,* também eles, devem ser seguidos como um todo. Não tem valor, por exemplo, a entrega de um produto atempadamente se o mesmo não tem qualidade suficiente e/ou vice-versa (princípios de entrega a tempo e integração de qualidade), e isto só é conseguido se, de certa forma, quer seja de uma forma mais ou menos rigorosa, todos os princípios forem aplicados.

2.2.2 Ferramentas, técnicas e métodos aplicáveis

Ferramentas *Lean*

Para alcançar os objetivos da implementação de *Lean* existe um conjunto de ferramentas, técnicas e métodos que facilitam o processo destacando-se, para o presente projeto as seguintes:

Análise da cadeia de valor:

O *Value Stream Mapping* (VSM) é uma ferramenta que permite, através do mapeamento de processos, identificar aqueles que não geram valor, rearranjar a sequência de fluxos numa mais adequada e identificar defeitos de qualidade (Hines e Rich 1997).

Gestão visual/ Controlo Visual:

Este tipo de ferramentas permite facilitar a comunicação e disponibilizar um conjunto de informações essenciais, tais como indicadores de desempenho, para a tomada de decisões conscientes baseadas em factos, permitindo que a informação não seja perdida (Imai 2012).

Sistema de controlo Kanban – *pull*:

As principais vantagens de um sistema Kanban são o alinhamento das tarefas a serem realizadas de acordo com as prioridades dos clientes, facilidade em perceber o estado de cada tarefa e o trabalho torna-se mais direcionado ao cumprimento de um objetivo específico (Poppendieck e Poppendieck 2003; Ahmad, Markkula, e Oivo 2013).

A utilização deste método encontra-se explorada na secção 2.1.2.

5 S:

Para alcançar os benefícios oferecidos pelas práticas *Lean* o uso de outros métodos/ferramentas indicado é o uso dos 5S *Seiri* (Organização), *Seiton* (Arrumação), *Seiso* (Limpeza), *Seiketesu* (normalização) e *Shitsuke* (autodisciplina) (Womack e Jones 2003). Importante destaque para o S *Seiketesu* dado que, para o âmbito do projeto em causa, poderá ser necessário analisá-lo mais atentamente, porque uniformizar os processos permite reduzir a variabilidade dos mesmos, eliminar etapas que não tragam valor diminuindo o seu tempo de realização (Dennis 2002).

PDCA:

O ciclo PDCA é uma ferramenta de melhoria contínua, representando cada uma das suas letras uma fase para implementar esta ferramenta (Sokovic, Pavletic, e Pipan 2010), são elas:

- Planear (*Plan*) – identificar problemas e desenvolver um plano de ação;
- Fazer (*Do*) – implementar o plano;
- Verificar (*Check*) – verificar os resultados da implementação;
- Agir (*Act*) – desenvolver ações corretivas, em caso de insucesso ou d não alcance dos resultados pretendidos.

5 Porquê:

O uso deste método tem como principal objetivo encontrar a causa para problemas, de modo a distinguir caraterísticas de um problema das suas causas. Assim ao encontrar-se um problema deve questionar-se, de modo sequencial, o seu porquê 5 vezes. Ohno, o criador deste método, definiu o número 5 como o sendo mais apropriado, mas não como sendo obrigatório.

Outras ferramentas

Matriz RACI:

Esta ferramenta permite atribuir e mapear as responsabilidades de cada processo aos seus envolvidos, permitindo classificar em: **R**esponsável (quem executa), **A**utoridade (responsável pela atividade), **C**onsultado (participa na decisão) e **I**nformado (quem recebe informações sobre o estado da tarefa).

Análise RICE:

O método RICE permite classificar os projetos de acordo com o seu alcance (***R**each*), o seu impacto (***I**mpact*), a confiança nas estimativas feita (***C**onfidance*) e no esforço necessário para realizar (***E**ffort*). Aplicando um cálculo simples estes 4 indicadores transformam-se num outro, RICE *score*, que permite identificar quais os projetos que devem ser realizados em primeiro lugar, de acordo com o resultado obtido (quanto mais alto o valor maior a sua prioridade) (McBride 2016).

$$\frac{\boldsymbol{R}each \times \boldsymbol{I}mpact \times \boldsymbol{C}onfidence}{\boldsymbol{E}ffort} = \boldsymbol{RICE}\ Score$$

2.3 Estratégias para integração de qualidade no produto através da eliminação de defeitos e assegurando a qualidade

IT *service management* (ITSM) é a gestão dos serviços de IT de uma ponta a outra para entregar aos clientes os serviços oferecidos, com base nas melhores práticas. Estas têm como objetivo dar suporte aos 4 processos mais importantes na gestão dos serviços de IT (Smith 2017). São estes (Smith 2017):

- Gestão dos pedidos de serviço – pedidos de material relacionados com software;
- Gestão de incidentes - pedidos relativos ao sistema estar em baixo, bugs;
- Gestão de problemas – mitigar problemas recorrentes que não podem ser prevenidos – bugs inesperados;
- Gestão da mudança – Método para controlar as mudanças necessário no *software* de forma a minimizar o impacto nos serviços, de forma uniformizada, ex.: *updates*, novas versões, pequenas/grandes melhorias etc.

A gestão deste processo pode ser feita através de um serviço denominado, comummente, por *Helpdesk* ou *Service Desk*. Este serviço pode ser construído sobre os princípios *Lean* de forma a tornar o processo, também ele, *Lean*, sendo importante a centralização deste serviço de modo a facilitar todo o processo de gestão. No capítulo 2.4 é feito um pequeno levantamento de alguns dos *softwares* que podem ser utilizados para este fim.

2.3.1 Como fazer a gestão de defeitos em desenvolvimento de *software* - *Helpdesk*

O *helpdesk* ou como também é conhecido *service desk* é um serviço central, nas organizações, onde os utilizadores, de um sistema, podem solicitar assistência (Siti-Nabiha, Thum, e Sardana 2012). As partes que o podem fazer são as equipas internas e relativamente a pessoas externas são os clientes, fornecedores e reguladores do governo (Lund 2012).

Um dos objetivos deste tipo de serviço é para resolver problemas da forma mais rápida possível, dando suporte ao utilizador num tempo adequado (Flynn 2014). Os problemas atrás

mencionados dizem respeito a eventos não planeados que podem levar à perda de negócio, frustração do utilizador entre outros (Flynn 2014).

Quem são os responsáveis por este serviço e quais as suas funções?

Num serviço de *helpdesk* os funcionários responsáveis por ele, dependendo da organização, podem ser externos, internos ou uma combinação entre ambos. Independentemente do seu papel face à organização estas pessoas são elementos chave e críticos uma vez que são elas um dos primeiros pontos de contacto entre cliente e organização (Siti-Nabiha, Thum, e Sardana 2012).

No que toca às funções especificas estas podem se dividir em 2 grupos, os funcionários aos quais lhes é dado o poder/conhecimento para resolverem alguns dos pedidos que surjam, sem necessidade de reencaminhamento do problema e ,por outro lado, os funcionários que apenas rececionam os pedidos de assistência e os reencaminham para quem dizem respeito (Datta e Lade 2015; Akinnuwesi et al. 2014).

Formas habituais de chegar a este serviço:

Existem inúmeras formas de obter ajuda através de um serviço semelhante a este, seja por contacto telefónico, mensagem de texto, *email*, preenchimento de formulário *online* entre outras (Akinnuwesi et al. 2014). Tome-se como exemplo algumas empresas especializadas em serviço de apoio ao cliente como a *Intercom* que utilizam a maioria dos meios de contacto atrás referidos, permitindo ao cliente começar uma conversa via chat para solicitar informações em tempo real.

De onde vem o valor do serviço de *helpdesk*?

O valor advém da eliminação da perda de produtividade pelos utilizadores não serem capazes de terminar as suas tarefas, devido aos problemas com que se depararam (Ashcraft et al. 2015).

Qual a importância do serviço de *helpdesk* para uma organização?

Quando um utilizador, de um determinado sistema de *software*, se depara com um problema que não o permite concluir as suas tarefas pode causar à organização, possuidora do *software* , a perda de clientes, reputação e até mesmo posição no mercado (Järveläinen 2013). Se uma falha no sistema pode ter todas estas repercussões, então pode-se concluir que é importante ter um bom serviço de *helpdesk/service desk.*

Também é importante que o sistema de *helpdesk* de uma empresa seja construído de forma a ter o menor impacto possível sobre a equipa de desenvolvimento, isto caso não exista uma equipa destinada apenas à resolução de erros. De acordo com Brian Bailey e Josep Konstan (2006) cada interrupção leva à necessidade de mais 3%-27% de tempo para a concluir do que sem interrupção, o que representa um grande impacto negativo nas normais tarefas desta equipa.

Conceitos relacionados a um serviço de *helpdesk*

O termo *ticket* é um conceito que, no âmbito dos serviços de *helpdesk*, significa uma dificuldade encontrada pelo utilizador que requer intervenção da equipa de desenvolvimento ou manutenção para a sua resolução (Datta e Lade 2015). Um *ticket* tem uma estrutura simples, possuindo um campo de texto onde vem a descrição do problema e/ou outras informações pertinentes. (Datta e Lade 2015).

Work-flow helpdesk

Tal como o processo de reportamento dos problemas pode ser feito de inúmeras formas o processo de triagem dos problemas também o pode. Existem diversas ferramentas onde os erros podem ser registados para análise desde plataformas *online* como o *Trello* (ver ponto 2.3) a ferramentas como o Excel. Após análise, estes podem ser reencaminhados para as equipas, com competências adequadas, os resolveram. Esta informação pode chegar a essas equipas através de inúmeras formas existindo plataformas desenhadas para esse fim (ver ponto 2.3).

2.3.2 Necessidade e ações a tomar para reduzir o número de erros reportados

Muitos autores já debateram e estudaram a eficácia de várias medidas alternativas para diminuir o número de erros reportados pelos utilizadores dos sistemas de *software*. É necessário ter algum cuidado, pois o que representa uma boa medida preventiva para uma organização poderá não o ser para outra.

Antes de se perceber quais as medidas que se podem tomar para melhorar este serviço é importante perceber o porquê da necessidade de o melhorar. Como já referido anteriormente, quando um utilizador tem um problema e necessita de assistência diminuiu a sua produtividade, o que representa perda de dinheiro; se diversos pedidos de *helpdesk* estão a ser formulados em simultâneo pode levar a que todos os recursos fiquem ocupados podendo impedir a resolução de todos os problemas com enfâse nos críticos; treinar os funcionários aptos para trabalhar em *helpdesk* consome muito tempo e é caro fazê-lo, entre muitas outras questões (Songsangyos, Niyomkha, e Tumthong 2012).

Com isto pode concluir-se que um sistema de *helpdesk* completo e bem estruturado poderá aumentar a eficiência e eficácia de uma organização pela diminuição do tempo de espera por uma solução bem como diminuição dos recursos gastos (humanos e monetários).

Medidas preventivas

Os defeitos, em *software,* podem ocorrer em qualquer fase do desenvolvimento e até mesmo após o lançamento de uma funcionalidade para o mercado. Quanto mais cedo se procede à sua correção menor serão os custos associados com o mesmo (Pandit e Tahiliani 2015).

Alguns das medidas já estudadas para a melhoria deste serviço passam por dar ao utilizador capacidade para resolver alguns dos seus problemas sozinhos. Alguns exemplos são a utilização de secções de *Frequent Asked Questions* (FAQ) onde o utilizador procura o seu problema e são mostradas instruções/dicas de resolução, tutorias (demos) entre outros tipos de soluções semelhantes (Songsangyos, Niyomkha, e Tumthong 2012). IBM, *Jira*, *Facebook* utilizam estes métodos. Outra forma de prevenir estes erros é submetendo o *software* a um conjunto de testes antes do seu *release*, para que se algum erro for encontrado este possa ser corrigido antes de ser passado para produção (Poppendieck e Poppendieck 2007; Hibbs, Jewett, e Sullivan 2009).

A utilização destas técnicas, FAQ, tutorias e mistura das duas, dá ao cliente poder para primeiro pesquisar as suas dúvidas e só depois, caso permaneça com dúvidas ainda, solicitar outro tipo de assistência mais adequada/completa, ou seja, são medidas prevenidas e que permitem educar os utilizadores, reduzindo a necessidade de estes solicitarem assistência, principalmente, para resolver erros derivados da má utilização do software (falta de conhecimento). É importante salientar que uma das maiores vantagens destes métodos é a não necessidade de intervenção humana. Estres métodos são utilizados por empresas como a IBM, *Facebook*, *Intercom* sendo que esta começou por ter sessões onde fazia demostrações de funcionalidades do seu produto online seguidas de uma sessão de Q&A (*Questions e Answers*). Rapidamente se apercebeu que a percentagem dos seus clientes que efetivamente atendia a estas sessões era muito baixa. Com estes baixos resultados decidiram terminar com este tipo de sessões e passaram a ter disponíveis tutorias para que os clientes os pudessem assistir sempre que quisessem e no horário que mais lhes conviesse. Esta mudança de abordagem melhorou bastante a taxa de visualização dos tutoriais e consequentemente aumento a taxa de aprendizagem das mesmas.

Para se perceber que método utilizar para a criação de uma secção para educar o utilizar sobre a utilização de *software*, quer de funcionalidades já existentes como de novas, bem como dar-lhe poder para resolver os seus problemas de forma independente sem recorrer ao pedido de assistência que poderá ser demorado, recorreu-se a estudos sobre quais os melhores métodos de aprendizagem. Os mais comuns na literatura são métodos com base no uso de texto, vídeo, imagens, texto e vídeo e até tutoriais interativos. Este último foi descartado porque o objetivo

desta secção era rever formas de aprendizagens fáceis e rápidas de implementar, com o menor uso de recursos possíveis.

Van Der Meij e Van Der Meij (2014) realizaram um estudo onde concluíram que, o uso de tutoriais em formato vídeo é mais eficaz do que o uso de texto. No caso do uso dos 2 em simultâneo obtiveram resultados muito próximos aos do uso de apenas vídeo. De seguida estudaram a retenção conseguida através do uso de cada uma das ferramentas atrás mencionada, verificando-se que o vídeo permitia uma maior retenção de conhecimento, seguido pelo uso conjunto de vídeo e texto (Van Der Meij e Van Der Meij 2014). Lloyd e Robertson (2012) chegou a uma conclusão semelhante, utilizando tutoriais vídeo e texto e aplicando-os a uma amostra, conclui que o uso de vídeo era vantajoso sobre o uso de apenas texto. A conclusão contrária chegou Mestre (2012) que concluiu que a melhor forma de apreender era através do uso de texto dando preferência ao uso de *bullet points* com palavras chave a *bold*. Tudo isto depende dos estilos de aprendizagem, dai as conclusões a que diversos autores chegaram serem diferentes e até mesmo contrárias, podendo as amostras ter uma grande influência sobre os resultados.

Antes de lançar uma nova funcionalidade para produção é necessário testá-la. Recorrer a um conjunto de testes com os critérios de aceitação corretamente definidos poderá evitar no futuro que surjam tickets de *helpdesk* a pedir a correção de certos erros.

Uma forma de corrigir estes erros é submetendo o *software* a um conjunto de testes antes do seu *release*, para que se algum erro for encontrado este possa ser corrigido antes de ser passado para produção (Poppendieck e Poppendieck 2007; Hibbs, Jewett, e Sullivan 2009).

Medidas corretivas

Como uma das principais medidas corretivas pode indicar-se a construção de um sistema de *helpdesk* bem organizado e robusto[4] (ver secção 2.3.1). Este implica a recolha de todos os dados pertinentes para a realização de um estudo intensivo sobre os erros reportados, de forma a encontrarem-se possíveis causas para os mesmos e assim proceder-se a um conjunto de medidas que permitam a mitigação/eliminação definitiva desses. A triagem dos problemas de acordo com um conjunto de categorias poderá ajudar também a esse propósito (Nakamura e Kijima 2011).

Simultaneamente a utilização de métricas adequadas, permitirá auxiliar todo este processo de identificação das causas bem como, posteriormente, avaliar melhorias após introdução de alterações de mitigação/eliminação dos problemas, de modo a permitir o melhor uso dos recursos disponíveis, nomeadamente da equipa *tech* (Akinnuwesi et al. 2014). Algumas das métricas sugeridas são: tempo até resolução, número de problemas atribuídos a cada elemento da equipa *tech*, número de *tickets* pendentes, nível de serviço, utilizador que reporta os erros (Akinnuwesi et al. 2014).

2.4 Plataformas de apoio à gestão de projetos

Este subcapítulo pretende explorar algumas das ferramentas, que existem no mercado, de apoio à gestão de projetos, com foco nos projetos de desenvolvimento de *software*. Em anexo é feita uma análise de vantagens e desvantagens das principais plataformas (anexo A).

[4] Um sistema de *helpdesk* bem organizado poderá não eliminar a raiz dos problemas, mas poderá ajudar a recolher e armazenar as informações necessárias para as encontrar, o que é de suma importância.

JIRA [5]

De acordo com "12th Annual State of Agile Report" (VersionOne 2018) uma das plataformas mais utilizadas para gestão de projetos de desenvolvimento *software Agile* é a plataforma *JIRA*.

O *JIRA* é uma plataforma online que permite, de forma organizada, gerir as diferentes iterações de desenvolvimento. Para isso permite enumerar e nomear cada iteração delimitando a sua duração. Em cada iteração são colocados *cards*, onde a tarefa é descrita, é atribuída uma prioridade (elevada, média, baixa…) e assignado um responsável bem como um vasto conjunto de outras opções tais como indicar o tempo espectável para concluir uma tarefa e posterior retificação do tempo real utilizado e identificação da fase de desenvolvimento. Esta plataforma permite ainda a divisão das tarefas em: Épicos (maior granularidade); Histórias; Problemas; Tarefas; Subtarefas (menor granularidade). Cada utilizador pode visualizar as suas tarefas, de iterações passadas, presentes e futuras (*backlog*). Podendo fazer a gestão das suas tarefas, arrastando-as, de acordo com a fase de desenvolvimento em que estão e pode ainda ver todas as tarefas da iteração quer lhe estejam ou não atribuídas. Isto representa uma grande vantagem pois permite que todos os membros da equipa tenham acesso à mesma informação e consigam trabalhar de forma colaborativa, tendo sempre em conta os objetivos pessoais, e mais importante os da equipa.

Microsoft Excel

De acordo com o estudo mencionado, a segunda plataforma mais utilizada é o *Microsoft Excel*. Esta ferramenta é usada para os mais diversos tipos de projeto, sendo fácil e intuitiva de usar. Apresenta como desvantagens a dificuldade de partilha de informação, isto é, não é colaborativa, não permitindo a atribuição de tarefas a indivíduos de forma fácil a não ser quando usada *online*, e neste caso perde bastantes funcionalidades. Uma das vantagens desta ferramenta é a facilidade com que se realizam estudos estatístico, sendo fácil calcular na mesma KPIs e outro tipo de estatísticas de controlo.

Slack [6]

O *Slack* é uma ferramenta que permite a troca de mensagens, ficheiros, videochamadas, a criação de canais de comunicação de grupo, por exemplo para discussão de projetos específicos, para discussão de problemas departamentais entre outros.

Trello[7]

O *Trello* é uma ferramenta que permite, a cada utilizador, pertencer a vários quadros (*boards*). Em cada *board* é possível criar um conjunto de colunas, as quais se podem nomear da forma que se entender. Posteriormente é possível criar cartões nos quais se podem descrever tarefas/ problemas, sendo possível adicionar a quantidade de membros que se achar conveniente, um conjunto de etiquetas personalizadas bem como deixar comentários. À medida que as tarefas vão sendo revistas é possível arrastá-las de uma coluna para a outra.

Outras

Existem ainda um vasto conjunto de ferramentas, como *VersioneOne, Bugzilla, Dropbox* e o *Google Docs* e o *Outlook*. Ferramentas mais dedicadas à gestão de *bugs Jira Service desk, Freshservice, Jira, MSP Manager, Instabug, Vision Helpdesk, FIT issue track*, e muitos outros, cada uma com diferentes características.

---

[5] https://www.atlassian.com/software/jira

[6] https://slack.com/

[7] https://trello.com/

3 Descrição e Caracterização do Caso de Estudo

O presente capítulo tem como objetivo apresentar e contextualizar o projeto numa situação real, explicando de uma forma detalhada o *core business* da empresa utilizada como caso de estudo.

Para ser possível realizar um projeto de melhoria primeiramente é necessário identificar os processos produtivos em causa, neste caso, de desenvolvimento de *software* para, posteriormente, se conseguirem identificar oportunidades de melhoria. Para isso, depois da contextualização e introdução do projeto, são detalhados os 3 processos que caracterizam o desenvolvimento de *software*, a saber: processos de especificação, processo de desenvolvimento e processo de manutenção de *software.*

3.1 Enquadramento

A HUUB é uma empresa que opera no campo da logística e gestão da cadeia de abastecimento. Fundada em 2015 tem como principal objetivo conseguir entregar a sua proposta de valor ao cliente – oferecer serviços logísticos à indústria da moda, com especial incidência em marcas de roupa para criança.

Esta empresa é muito mais do que aquilo que a sua proposta de valor faz parecer. Ao conhecer--se a sua missão e visão rapidamente se compreende isso mesmo e se percebe a real dimensão daquilo que tem para oferecer ao mercado. O seu valor, enquanto empresa, surge nos serviços de valor acrescentado que oferece, suportados pelo uso de tecnologia desenvolvida internamente, uma plataforma digital. A missão da HUUB passa pela simples premissa de *Connect everything*, quer isto dizer que esta tem como principal objetivo ser o centro de um ecossistema dinâmico, ligando clientes (marcas), utilizadores finais (clientes das marcas), fornecedores e parceiros, através da ligação da cadeia de valor ponta-a-ponta, tal como a Figura 5 ilustra. Com isto a HUUB pretende ajudar os seus clientes a crescer, para que, à medida que eles crescem, ela possa crescer com eles.

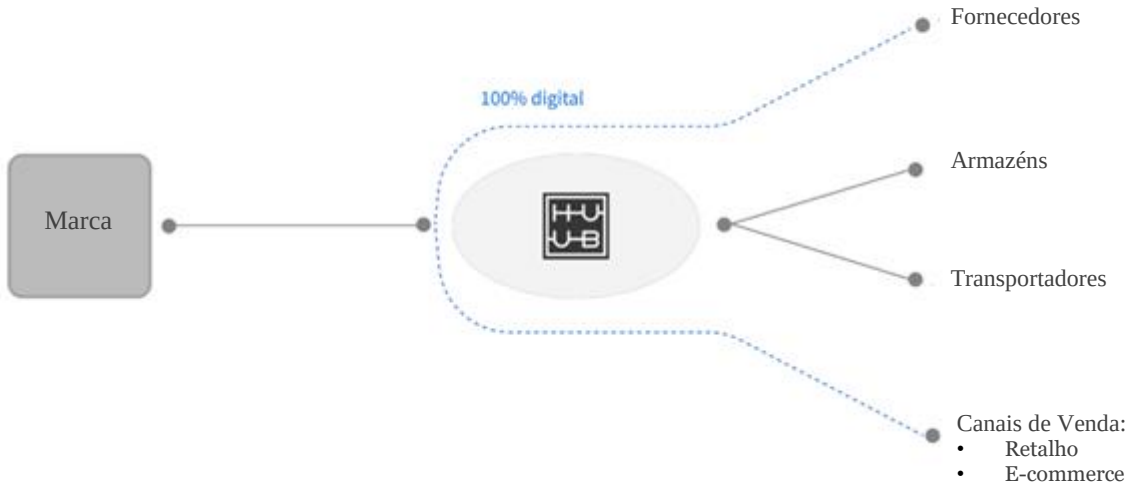


Figura 5 - Posicionamento da Huub na cadeia de abastecimento

Para se entender o propósito e pertinência deste projeto é fundamental conhecer-se de forma detalhada e objetiva a proposta de valor oferecida por esta empresa e assim perceber qual a relevância e impacto do mesmo para esta.

A empresa gere, para cada cliente, todos os processos de logística desde os fornecedores até ao cliente - lojas físicas ou mesmo consumidor final - oferecendo um preço único por cada peça movimentada. Isto representa, para os seus clientes, uma enorme simplificação na gestão de contas, comparativamente à proposta de outras empresas. Para conseguir que este fluxo de produtos funcione da forma desejada, a HUUB possui um conjunto de armazéns – próprios e

externalizados - onde os itens, vindos dos fornecedores, são organizados e posteriormente embalados para serem distribuídos para o cliente final.

Através da gestão da cadeia de valor, de cada um dos seus clientes, a empresa fica na posse dos mais variados tipos de informação, como por exemplo o número de itens vendidos, os tipos de itens mais e menos vendidos e as suas caraterísticas. Estas informações são uma mais valia, pois, uma vez analisadas, poderão ajudar as marcas a registar tendências de consumo dos seus clientes e assim crescer.

Todos os fluxos de informação, referentes ao cliente, são geridos através de uma plataforma única, criada de raiz pela empresa. A plataforma foi desenvolvida para ser usada pelos clientes e pelas equipas internas (*account managers* (AMs), equipa de operações, entre outros elementos). Nesta são registados todos os processos físicos feitos em armazém (*picking*, *labeling*, entre outros) e todos os processos feitos virtualmente (registo de clientes, controlo dos fornecedores de cada cliente, gestão de encomendas, entre outros), representado o *input* crítico e essencial ao correto funcionamento de todas as operações, para conseguir entregar ao cliente os serviços oferecidos.

Assim, é facilmente percetível que o funcionamento de toda a empresa gire em torno do funcionamento desta plataforma, sendo fundamental que este *software* garanta a satisfação de todas as necessidades, quer dos clientes, quer das equipas internas.

3.2 O Projeto

Como referido anteriormente, o sucesso e valor acrescentado da oferta da HUUB, no mercado, depende do bom e correto funcionamento da plataforma por si desenvolvida. Esta afirmação torna-se ainda mais verdadeira ao não existirem no mercado soluções de *software* completas que permitam satisfazer as necessidades de gestão, de uma ponta à outra da cadeia de valor, de cada um dos seus clientes.

O enfoque do presente projeto passa pela análise de todo o processo de especificação, desenvolvimento e manutenção de *software* para identificação e implementação de possíveis melhorias do mesmo.

Esta necessidade surge do facto de a empresa estar a utilizar uma primeira versão da plataforma, que foi apenas criada para resolver um conjunto de problemas, não conseguindo a mesma acompanhar o elevado crescimento que a empresa está a ser alvo de. Com isto, pode dizer-se que a plataforma não tem escalabilidade, o que acaba por subcarregar os *account managers* devido a uma maior dificuldade em utilizar as funcionalidades. Assim é urgente a criação de uma nova versão da plataforma e, para isso, é fulcral que os processos de especificação e desenvolvimento desta, bem como a manutenção da atual plataforma e, posteriormente, a manutenção da nova sejam melhorados de modo a acelerar o processo. Importante notar que à data de início do projeto a empresa se encontrava já a desenvolver a nova plataforma sentindo uma premente necessidade, mas também uma grande dificuldade em acelerar o processo de construção desta. A necessidade está relacionada ao crescimento esperado, devido a um grande investimento, e consequente abertura de uma janela para o fazer dentro de um espaço curto de tempo, não dando à plataforma atual possibilidade para sustentar esse crescimento; a dificuldade deve-se à constante manutenção que a plataforma em uso necessitava e que era fulcral para a continuação e realização de todas as operações internas e gestão de clientes.

Esta urgência advém ainda da necessidade da empresa reunir as condições necessárias para crescer, mantendo e adquirindo novos clientes, bem como de permitir aos seus utilizadores um uso autónomo e assim conseguir oferecer outros serviços, como de consultoria à gestão da marca que até ao momento têm sido deixados em *stand by*.

3.3 Caracterização "AS IS"

Como já referido, a empresa oferece serviços de valor acrescentado suportados por uma tecnologia evolutiva desenvolvida internamente de modo permanente e incremental, deparando-se com diversos desafios: desenvolver uma nova plataforma de forma mais eficaz e eficiente e manter a atual com a máxima eficiência e eficácia no contexto das atividades corretivas do sistema, até entrada em funcionamento da nova plataforma. Este subcapítulo tem o objetivo de dar a conhecer o modo de funcionamento, à data de início do projeto, das metodologias usadas nas atividades de especificação, desenvolvimento e manutenção na gestão de desenvolvimento de *software*, para que se entenda a raiz das dificuldades e para que faça sentido a identificação dos problemas no capítulo 4 e posterior identificação de soluções propostas no capítulo 5.

3.3.1 Visão geral sob os Processos envolventes na criação de *software*

De um modo geral o processo de desenvolvimento de *software* pode ser dividido em três grandes fases, como mostra a Figura 6, sendo todo o processo detalhado através de uma matriz de responsabilidades na Figura 7, no fim do presente capítulo.

Figura 6 – Processos de desenvolvimento de *software*

Para que o *software* desenvolvido tenha uma boa qualidade é necessário garantir que as três fases do processo sejam executadas da forma mais eficiente e eficaz possível. De acordo com a metodologia selecionada o processo de passagem para cada uma destas fases poderá ser linear ou iterativo. Para se entender o que compreende cada uma das mesmas estas são especificadas detalhadamente nos subcapítulos seguintes.

3.3.2 Processo de Especificação de *Software*

A especificação do produto é uma das partes cruciais aquando do desenvolvimento tecnológico sendo uma das primeiras etapas a ser feita. Se esta, por algum motivo, é feita de uma forma incorreta ou menos adequada, todo o restante processo poderá ficar comprometido.

Na empresa em estudo, primeiramente, procurava-se entender as necessidades dos *stakeholders* (internos e externos) através de reuniões com os mesmos para levantamento de possíveis requisitos. De seguida, passava-se para a especificação escrita e visual do resultado dessas reuniões. A responsabilidade de execução destas tarefas cabia à equipa de produto, não só do levantamento de requisitos, mas também da especificação escrita e visual (interface) das funcionalidades a desenvolver. As ferramentas de suporte para este processo utilizadas eram a *Dropbox Paper* (especificação escrita) e o *Figma* (especificação visual).

Para a especificação escrita não existia uma *framework* e/ou *guideline* de orientação, cabendo, a cada elemento a organização do documento da forma que considerasse mais adequada. Relativamente à especificação visual o processo era uniformizado, existindo um conjunto de elementos pré-definidos que eram usados na construção dos *mockups*. Esta informação encontrava-se descentralizada e dispersa pela não integração da mesma num documento único.

3.3.3 Processo de Desenvolvimento de Software

Após a especificação de requisitos inicia-se o processo de desenvolvimento de *software*. Para uma correta gestão do mesmo é necessário primeiro que se defina a metodologia de desenvolvimento a utilizar e, consoante o tipo de metodologia escolhido, planear o desenvolvimento.

No início do projeto a empresa usava como metodologia de desenvolvimento *Agile*, mais concretamente, aproximava-se do método *Scrum*, mas ainda sem usufruto de todos os seus benefícios e utilização de práticas na totalidade, estando numa fase ainda de ajustamento a este método. A gestão de todo este processo era dirigida pela equipa de produto com apoio da equipa de desenvolvimento.

Como explicado no capítulo 2.1.2, o uso desta metodologia pressupõe a utilização de um conjunto de práticas (Schwaber e Sutherland 2013), algumas delas usadas já pela empresa:

Sprint

O tempo de duração de cada *sprint* era de 2 semanas. Para a gestão dos *sprints* era utilizada uma ferramenta de apoio para a alocação e controlo das tarefas, o *JIRA*, utilizando-se conjuntamente com as duas ferramentas usadas para especificação, *Dropbox paper* e *Figma*. No que toca à gestão dos *sprints* e posteriormente de *releases* não existiam ferramentas nem métricas de controlo bem definidas.

***Sprint Planning*:**

A primeira segunda-feira de cada *sprint* era destinada à discussão de problemas, desbloqueio de *bottlenecks*, priorização e divisão de tarefas. Nestas reuniões estavam presentes a equipa de produto (quem organizava e dirigia as reuniões) e a equipa *tech*. A duração destas era variável, mas em média era de duas horas, chegando por vezes a ocupar uma manhã inteira.

O *output* esperado da reunião era a alocação de tarefas a cada elemento da equipa *tech* e esclarecimento de dúvidas relativas às mesmas havendo ainda lugar para a discussão de alguns problemas. A alocação das tarefas era feita através da plataforma *JIRA*, servindo também para controlar o fluxo e o estado de execução das mesmas. A alocação das tarefas a cada elemento era feita não só consoante área de especialização, mas também consoante a disponibilidade dos mesmos. Para isso a cada tarefa eram dados *story points* [8] sendo a atribuição destes valores arbitrária e feita pelo elemento da equipa *tech* responsável. A disponibilidade de cada elemento era também convertida em *story points* para que à medida que as tarefas iam sendo alocadas o número de *story points* de cada tarefa ia sendo subtraído da disponibilidade de cada elemento. Ao chegar a zero ou a valores negativos a atribuição era terminada. Caso se verificasse que, devido a dependências de tarefas de outros elementos, era necessário colocar mais algumas tarefas em *sprint* verificavam-se quais as que se poderiam retirar de acordo com o mesmo critério.

***Daily meeting*:**

Esta reunião realizava-se todos os dias às 10 horas, à exceção do primeiro e último dia de cada *sprint*, em que não se realizava. Nesta reunião estavam presentes todos os elementos da equipa *tech* e de produto e, duas vezes por semana, estavam ainda presentes, *stakeholders*.

Esta reunião era realizada em torno do ecrã de um computador num local com pouco espaço e visibilidade. A reunião era dirigida pelo *Product Owner* e cada um dos elementos da equipa

[8] Story points são uma unidade de medida que permitem classificar uma user story com uma pontuação de 1, 3, 5, 8 ou 13 sendo que 1 representava uma tarefa pouco complexa e/ou demorada e 13 uma tarefa muito complexa e/ou demorada.

tech ia expondo alguma da informação que considerasse relevante, como por exemplo a partilha de informações sobre o que tinha sido produzido no dia anterior, levantamento dos problemas com que se estavam a deparar dando lugar para a discussão dos mesmos e ainda comunicação do que iriam estar a trabalhar naquele dia.

Estas reuniões foram introduzidas com o objetivo da equipa *tech* partilhar o estado do seu trabalho, não só devido às dependências que pudessem existir com as tarefas de outros elementos, bem como para comunicarem problemas com que se iam deparando, esperando receber *feedback* dos presentes na reunião.

Quando na presença de *stakeholders* era dado lugar para que estes interviessem, o que geralmente resultava no pedido de esclarecimento de informações relativas ao que estaria a ser discutido, bem como de problemas reportados e data de resolução esperada para resolução dos mesmos.

***Sprint Reviews*:**

Estas reuniões aconteciam na última sexta-feira de cada *sprint*. Eram iniciadas com o *Product Owner* a fazer uma breve revisão do que havia sido o planeado para o *sprint*. De seguida, o mesmo partilhava, por vezes, a taxa de execução do *sprint* e iniciava o diálogo com a equipa de desenvolvimento de modo a avaliar todo o processo. Nesse momento cada elemento revia todo o *sprint* fazendo o seu ponto de situação do mesmo e partilhando o estado de cada tarefa e as dificuldades sentidas, o que abria espaço à discussão de possíveis soluções. Nesta reunião estavam ainda presentes *stakeholders* que assumiam uma postura semelhante à das *daily meetings*. Estas reuniões duravam em média 3 horas. O que era chamado de *sprint review* era na realidade uma mistura entre o *sprint review* e o *sprint retrospective* (Schwaber e Sutherland 2013).

3.3.4 Processo de Manutenção de *Software* - Atividades corretivas

Tal como definido por Flynn (2014), um dos principais objetivos de um serviço de *helpdesk* é o de resolver problemas da forma mais rápida possível dando suporte ao cliente atempadamente. Foi com este propósito que a empresa criou este sistema, como tentativa de centralizar a comunicação de problemas relativos à utilização da plataforma 1.0, apenas acessível a utilizadores internos para que, deste modo, conseguissem gerir a manutenção necessária a ser feita à plataforma.

O fluxo de comunicação criado para o *helpdesk* era complexo e desorganizado, podendo ser feito de inúmeras formas alternativas, desde utilização de plataformas digitais a contacto pessoal, tendo todas elas o mesmo *trigger* – problema encontrado pelo *Stakeholder* – sendo toda a equipa de desenvolvimento responsável por assegurar este serviço.

Para tornar este processo mais eficaz, pouco tempo após a criação deste serviço de ações corretivas de apoio ao cliente, foi criada a figura do "bombeiro". Esta figura correspondia a um elemento da equipa *tech* que tinha a responsabilidade de ir vigiando os vários canais de comunicação (*Slack, Trello, Outlook*, contacto pessoal) e resolvendo os problemas que iam surgindo e/ou redirecionando-os para o membro da equipa mais especializado na área em causa e/ou disponível. A personagem do "bombeiro" era rotativa mudando a cada *spri*nt.

Até à data de início do projeto, não existiam ferramentas, como *templates* e/ou *guidelines,* que permitissem aos *stakeholders* expor os seus problemas de forma rápida e organizada. A forma de comunicação entre os *stakeholders* (internos) e o responsável pelo *helpdesk*, no *sprin*t em questão, era feita através do uso de uma das três plataformas mencionadas na Tabela 4 bem como os respetivos casos de uso.

Tabela 4 – Plataformas de comunicação, respetivos caso de uso e vantagens e desvantagens de utilização

	Casos de utilização			Procedimento	Vantagens(v) Desvantagens(d)
Slack	B	M	E	Comunicação via canal: bombeiro. Mensagem de texto sem pré formatação	(v) Tomada de conhecimento rápida (v) Rápida troca de informações (d) Perda da informação pela não existência de histórico
	X	**X**	**XX**		
Trello	B	M	E	Comunicação no *board: helpdesk.* Criação de um ticket, sem pré formatação, com possibilidade de colocação de *labels* pré-definidas[9]. Distribuição dos tickets por colunas: *Backlog, WIP* e *Done*.	(v) Visualização de informação de acordo com o estado de desenvolvimento do problema.
	X	**X**	**X**		
Outlook	B	M	E	*E-mail* sem pré formatação	(d)Troca de informações demorada (d) Perda de informações
	X				

(Prioridade B -Baixa, M -Média, E – elevada)

O contacto pessoal era feito de forma constante, qualquer que fosse a urgência do pedido, principalmente quando *os stakeholders* não obtinham uma resposta dentro do limite de tempo que considerassem aceitável. No anexo B o fluxo é detalhado de acordo com o processo e a entidade responsável por o executar. No caso particular do erro/problema ser levantado pelos *stakeholders* externos o procedimento era diferente. Estes contactavam o AM responsável via telefone ou email e este ou resolvia o problema, no caso de este derivar da falta de conhecimento do funcionamento da plataforma, ou quando fugia dos seus conhecimentos o AM reportava-o através do serviço de *helpdesk* acima descrito.

[9] Apesar de as *labels* serem pré-definidas não existia uma explicação do significado de cada uma, cabendo a cada um o uso dos *labels* de acordo com a interpretação que fazia dos mesmos. As opções possíveis eram: Pouco Urgente; Urgente; Muito Urgente; Cliente; Erro: Urgente; Erro: Análise e por fim Integração

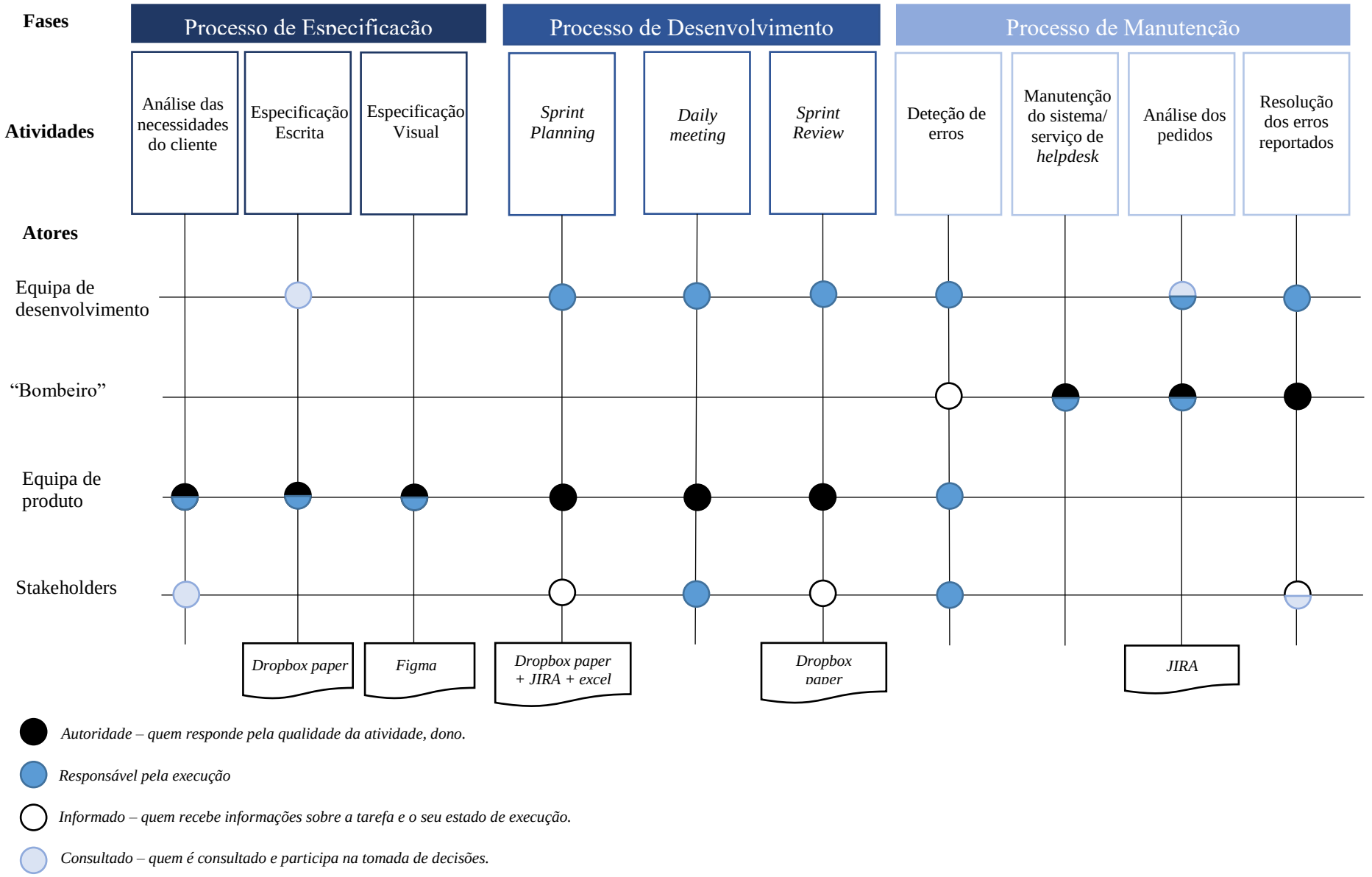


Figura 7 - Matriz RACI dos 3 processos envolvidos no desenvolvimento de Software, "AS-IS".

4 Identificação de Problemas

Este capítulo tem como objetivo apresentar a metodologia seguida para a identificação de oportunidades de melhoria, para no capítulo 5 serem exploradas as ações adequadas a tomar de acordo com o exposto no presente capítulo, e dar assim resposta ao objetivo O1.

Estando o projeto enquadrado numa empresa *early stage*, onde a maturidade do produto tecnológico ainda não era devidamente robusta, era complexa a obtenção de dados ou métricas que permitissem analisar o estado inicial dos processos aquando do início do projeto. Para a identificação de oportunidades de melhoria (problemas) e perceção do estado real dos processos foi necessária, primeiramente, a recolha de dados simultaneamente à observação do desenrolar das atividades e em alguns casos, a participação nas mesmas. A análise de dados permitiu, através da introdução de algumas métricas de controlo adequadas (KPIs), identificar as potenciais oportunidades de melhoria.

A identificação e procura de problemas teve como fio condutor as metodologias e princípios já referidos ao longo do capítulo 2 e que são, grosso modo, os 7 princípios *Lean* de Mary e Tom Poppendieck (2003, 2007), no caso específico do processo de desenvolvimento de *software* para além deste fio também se teve em atenção os métodos de desenvolvimento de *software* e no caso do processo de manutenção, para além dos dois fios orientadores já expostos utilizou-se um terceiro sobre técnicas de integração de qualidade.

4.1 Problemas no Processo de Especificação

A equipa, durante o desenrolar de todo o projeto, encontrava-se a desenvolver a nova versão da plataforma. Como tal era essencial identificar os problemas e causas que afetavam o processo de especificação, dada a sua enorme importância, e que haviam sido cometidos no desenvolvimento da plataforma atualmente em vigor, para evitar que fossem também cometidos na construção da nova versão. Para isso os processos foram analisados tendo por base os conceitos *Lean* explorados no capítulo 2.2 tendo-se utilizado, mais concretamente o método dos 5 porquê para identificar os problemas e as suas causas.

A análise e participação no processo de especificação de requisitos permitiu identificar os seguintes problemas e consequentes causas:

- **(P1) Necessidades dos utilizadores diferentes das funcionalidades desenvolvidas**

A interface da plataforma não é visionada da mesma forma pelos diferentes tipos de utilizadores (internos e externos). A interface a que os clientes têm acesso é diferente do das equipas internas. Do ponto de vista dos clientes a plataforma tem alguma falhas uma vez que não disponibiliza funcionalidades que lhes permitam fazer as ações de que necessitam de uma forma rápida e intuitiva, tal como abordado no capítulo 3.2 com mais detalhe. Este problema advém não só da baixa manutenção feita a esta interface, ver capítulo 4.3, mas também de uma especificação incorreta de requisitos e consequente desenvolvimento de uma plataforma diferente da desejada/útil para o cliente. Este problema é identificado como um desperdício – funcionalidades desnecessárias – de acordo com Mary e Tom Poppendieck (2003, 2007).

As causas identificadas para este problema são:

- (C1.1) Especificação de requisitos incompleta;

O levantamento de requisitos era feito apenas através de uma recolha de informações inicial, não se confrontando o cliente com o resultado desse mesmo levantamento de requisitos, por exemplo através dos protótipos criados entre outras possíveis técnicas, o que levou a que a plataforma desenvolvida fugisse às necessidades dos mesmos em algumas funcionalidades.

No caso dos *stakeholders* internos o problema era semelhante, mas, devido a um maior uso da plataforma e maior proximidade entre os utilizadores e os responsáveis pela criação da mesma, esta era mais cuidada e, consequentemente, mais próxima dos requisitos, apesar das falhas que possuía, derivadas da especificação incompleta.

- (C1.2) Processo de especificação não uniformizado e consequente dificuldade de especificação das funcionalidades desejadas;

Como o processo não era uniformizado, cada vez que se queria especificar uma nova funcionalidade criava-se um documento onde se escrevia um conjunto de critérios e como não existia uma *framework*, por vezes a informação inserida estava incompleta e/ou redundante e a linguagem utilizada era distinta de um documento para o outro, resultando assim também em diferenças entre a especificação escrita e visual e numa maior dificuldade em compreender os requisitos.

- **(P2) Não utilização simultânea da especificação escrita e visual bem como defeitos e incoerências entre os mesmos.**

Por vezes, a descrição dos elementos visuais não correspondia aos desenhados devido à falta de coerência entre o nome atribuído a um determinado elemento na ferramenta visual e na especificação escrita. Isto levantava dúvidas e em alguns casos a construção de funcionalidades diferentes das desejadas, o que posteriormente levava à necessidade de correção das mesmas (retrabalho). Este problema é também uma das causas do problema anteriormente e é identificado como um desperdício – processos desnecessários – de acordo com Mary e Tom Poppendieck (2003, 2007).

Para este problema a principal causa identificada foi:

- (C2.1) Dispersão da informação

A equipa acabava por usar a especificação escrita ou visual em vez das duas em simultâneo devido à não compactação da mesma em local único.

Importa notar que os problemas e causas identificadas no processo de especificação estão também, maioritariamente, na origem e base dos problemas do processo de desenvolvimento e manutenção de *software* discutidos nos capítulos 4.2 e 4.3.

4.2 Problemas no Processo de Desenvolvimento

Para se entender o estado do processo de desenvolvimento fez-se a recolha dos dados históricos de cada *sprint* (informações registadas no *JIRA*) e, posteriormente, procedeu-se a uma análise detalhada e subsequente introdução de métricas de controlo. A análise dos dados conjuntamente com a análise dos processos, tendo por base os conceitos *Lean* (cap.2.2) e os métodos de desenvolvimento de *software* (cap.2.3), permitiu identificar os seguintes problemas, bem como as suas causas:

- **(P3) Taxas execução de *sprints* irregulares e tendencialmente baixas:**

Para o cálculo da taxa de concretização foram retirados todos os *tickets* referentes a *helpdesk*, sendo a análise dos mesmos feita separadamente (ver capítulo 4.3). Após a filtração destes dados cada uma das tarefas foi classificada consoante a sua conclusão, não conclusão ou remoção no *sprint.* Com estes dados calculou-se a taxa de execução dos mesmos (métrica de controlo implementada), obtendo-se resultados extremamente irregulares, tal como é possível observar na Figura 8. Importante notar que após uma análise mais detalhada concluiu-se que a taxa obtida no "*sprint* extra" é enganosa uma vez que 88% das tarefas executadas foram tarefas começadas nos *sprints* anteriores, o que não aconteceu em mais nenhum dos *sprints*, e

consequentemente levou a que o tempo necessário para concluir as tarefas fosse mais pequeno. Este problema é identificado com um desperdício – tempo de espera - uma vez que taxas baixas representam um maior tempo de espera para o lançamento de funcionalidades (Poppendieck e Poppendieck 2003, 2007). Pode ainda ser identificado como o não cumprimento do 2º princípio Lean – entrega rápida - de acordo com a ordem com que os mesmos foram expostos no capítulo 2.2.1.

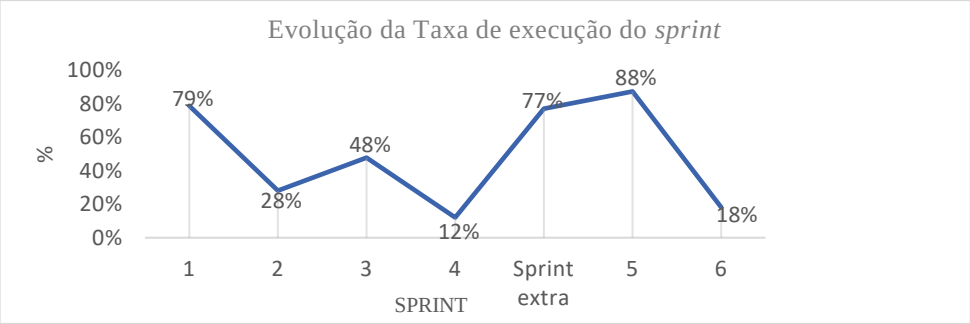


Figura 8 - Evolução da taxa de execução

Foram identificadas algumas das possíveis causas para estes valores, que são:

- (C3.1) Não existência de métricas de conhecimento da evolução da equipa no fim de cada *sprint*.

Nas *sprints reviews* nem sempre era partilhada nem calculada a taxa de execução, a não partilha/ cálculo da mesma levava ao desconhecimento da equipa do real estado de cada *sprint*. Assim, transportava-se para o seguinte as tarefas não concluídas no mesmo, não se investigando as causas para que nuns *sprints* a quantidade de tarefas a transportar fosse maior (taxas mais baixas) ou menor (taxas mais altas), o que não incentivava à tomada de ações de melhoria nos *sprints* que corriam menos bem.

- (C3.2) *Sprints* de 2 semanas;

O período temporal de cada *sprint* foi sempre de duas semanas não se tendo exposto a equipa a *sprints* com diferentes durações de modo a determinar qual o período ótimo para a mesma.

- (C3.3) Falta de visualização de dependência entre tarefas;

A gestão de todas as tarefas, tal como referido no capítulo 3.3.3, era feito exclusivamente através do *JIRA*, o que não permitia ter uma visão global das dependências entre tarefas bem como não permitia determinar qual o estado de conclusão de cada épico e *user story* consoante o número de tarefas que a mesma tinha.

- (C3.4) Dificuldade em perceber e quantificar *story points* o que resulta num planeamento inadequado;

Através da observação da Figura 8, é possível constatar que quando a taxa atingida era alta no *sprint* imediatamente seguinte a taxa descia bruscamente. Isto poderá ser explicado pelo facto de os *sprints* com taxas de execução elevadas serem *sprints* bem planeados, *story points* bem ajustados (*sprint* 1 e 5) e/ou *sprints* onde são fechadas as tarefas incompletas dos *sprint*s anteriores (*sprint* especial), que por já estarem adiantadas levam menos tempo a terminar. Outra justificação poderá ser o facto de que quando se obtinham taxas muito altas aumentava-se a carga da equipa de uma forma demasiado ambiciosa não se conseguindo cumprir nem 50% do proposto (*sprint* 2,4 e 6).

- (C3.5) Existência de imprevistos não contabilizados;

Uma das possíveis causas para as taxas baixas e irregulares poderá advir do facto de terem existido mais imprevistos nesses *sprints* como reuniões, *grommings* e outros pedidos não

contabilizadas na disponibilidade da equipa e sem um registo correto dos mesmos. A interrupção de tarefas, tal como revisto no capítulo 2.3, poderá aumentar até 27% o tempo de execução de uma tarefa (Bailey e Konstan 2006) e, consequentemente, diminuir a taxa de execução de um *sprint*.

- (C3.6) Especificação de requisitos sem uniformização

Os *sprints* com taxas mais baixas podem ser justificados ainda pela dificuldade em desenvolver funcionalidades de acordo com a especificação, devido às falhas que esta possui-a (ver capítulo 4.1), o que posteriormente levava à necessidade de retrabalho, não sendo dada como completa a tarefa. Assim, para cumprir os requisitos reais, depois de ser produzida, a tarefa tinha de ser refeita logo. O tempo perdido com uma mesma tarefa era superior ao estimado, o que resultava em taxas baixas.

- (C3.7) *Backlog* do *sprint* extenso e não priorização do mesmo

A quantidade de tarefas que se ia acumulando, tal como se observa na Figura 9, aumentava de *sprint* para *sprint* o que representa um desperdício de acordo com Mary e Tom Poppendieck (2003, 2007) e Hibbs, Jewett e Sullivan (2009). Um *backlog* extenso causa a permanente sensação de não finalização do trabalho o que poderá ter um impacto negativo.

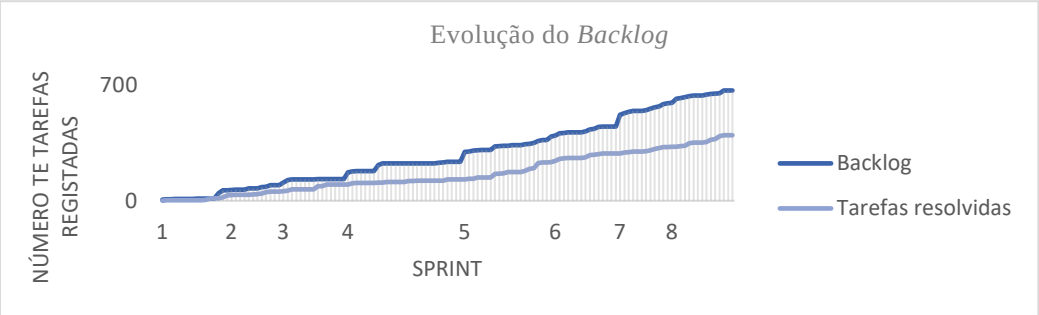


Figura 9 - Evolução do backlog ao longo dos sprints

- (C3.8) Equipa pouco *cross-functional;*

Em cada *sprint* as tarefas eram distribuídas por áreas de especialização, ou seja, cada membro focava-se apenas na concretização de tarefas dentro da sua área de conhecimento. Quando algum dos elementos não conseguia executar todas as suas tarefas e como os restantes sentiam dificuldade em realizar tarefas fora da sua área de conhecimentos não as efetuavam quando terminavam as suas, o que poderá também ser uma das possíveis causas para as taxas baixas.

- **(P4) *Daily meetings* longas e sem cumprimento do seu objetivo:**

Para se perceber o modo de funcionamento das *daily meetings* estas foram observadas durante 1 *sprint* e meio (parte do 6 e 7 completo), registando-se a duração de cada reunião bem como o número de intervenientes na mesma. Após o estágio de observação constatou-se que a duração média das reuniões era de 31.52 minutos o que de acordo com Schawber e Sutherland (2013) representa uma duração de 113% acima do tempo ideal, uma vez que estas devem durar cerca de 15 minutos.

Foram identificadas algumas das possíveis causas para estes valores, a saber:

- (C4.1) Pouca visibilidade sobre o que estava a ser discutido;

Como a discussão era feita em torno de um monitor de computador pequeno, não estando visível para todos os membros, existia dificuldade em acompanhar a discussão e era perdido tempo a

passar a informação de uns elementos para os outros, o que pode também implicar a introdução de ruído na comunicação e a transmissão da informação pode ser afetada.

- (C4.2) Não existência de *framework* (processo não uniformização)

A reunião não seguia uma estrutura bem definida, não sendo respeitada a estrutura das três perguntas tal como proposto por Schawber e Sutherland (2013), o que alongava o discurso dos participantes e abria lugar à discussão de assuntos não oportunos à reunião e, consequentemente, a duração da mesma também aumentava.

- (C4.3) Presença de *stakeholders*

Na presença de *stakeholders* as reuniões tornavam-se mais longas, uma vez que estes intervinham solicitando informações sobre resolução de pedidos feitos à equipa *tech* e, devido à linguagem técnica e específica usada nestas reuniões, eram pedidas, por vezes, explicações em linguagem corrente, o que prolongava a duração da mesma.

4.3 Problemas no Processo de Manutenção

Como forma de se compreender o real impacto que o processo de *helpdesk* estava a ter na empresa em geral, bem como o funcionamento do mesmo, realizou-se uma análise do processo e dos dados, exaustiva. Da observação do mesmo, com base nos conceitos *Lean* (2.2) foram identificados os seguintes problemas:

- **(P5) Informação dispersa e pouco visível**

A causa identificada para este problema é:

- (C5.1) Utilização de diversas plataformas:

Os problemas eram registados em diversas plataformas o que levava à perda e dispersão da informação, não só pela não existência de histórico em *Slack* (uma das plataformas mais frequentemente usada para reportamento de erros) bem como da informação retida nas contas pessoais. Este mesmo problema acoplava um outro, a dificuldade em fazer rastreamento (*tracking*) e consequente visibilidade de ambas as equipas (a que reporta e a que resolve) sobre os problemas e o seu respetivo estado de resolução. Este problema dava origem ao esquecimento de determinados problemas, a sua não resolução e a construção de outras funcionalidades em cima de *bugs*.

Para ser possível continuar com a análise do processo, e dada a não existência de registo de muitos dos problemas levantados e a não compactação em local único dos problemas registados, procedeu-se imediatamente a ações corretivas de recolha e análise de informação. Primeiramente compactou-se toda a informação, considerada como relevante, relativa ao *helpdesk*, num único local onde, e para cada problema reportado, recolheram-se as seguintes informações: *sprint* em que foi reportado; *sprint* em que foi resolvido; nome; prioridade; identidade de quem reportou; identidade de quem resolveu; tempo gasto (desde a análise até finalização do *ticket*); descrição do problema; outras notas; data de criação; data de finalização.

De seguida, e como forma de trabalhar estes dados, criaram-se categorias de erros para posteriormente se proceder à classificação de cada um dos problemas individualmente, o que foi feito com base no aconselhado por Nakamura e Kijima (2011). Este processo foi realizado com o objetivo de se conseguir compreender e identificar a causa dos problemas e fazer uma separação entre os tipos de problemas apresentados. Para este fim criaram-se 4 categorias de erros exposta na Tabela 5.

Tabela 5 - Categorização de cada tipo de erro

Categoria	Descrição	Sub Categoria	Descrição
Tecnológico (Manutenção)	Erros diretamente imputados a um incorreto funcionamento da plataforma.	*Bugs*	Erro, defeito ou falha no sistema de informação que produz efeitos não previstos. A fonte do problema reside em problemas no código.
		Externo	Problema que produz efeitos não previsto devido a problemas externos, ex.: *internet* em baixo, máquinas em baixa, alterações de plataformas *online* do lado dos clientes.
Processo	Falhas na execução do processo na plataforma	Complexo manualmente	Realização do processo via manual (plataforma) é demasiado longa/complexa inviabilizando a sua execução.
		Não conhecimento	Desconhecimento do processo por parte dos *stakeholders* ou falha/erro na execução do mesmo.
Sugestão	Processo funcional, na plataforma, mas os *stakeholders* sugerem formas de o tornar mais fácil e rápido de acordo com a sua experiência e perceção do processo.		
Limitação	Necessidades da parte dos *stakeholders* e que não existem *features* para satisfazer essa necessidade na plataforma.		

- **(P6) Tempo até resolução de problemas muito elevado:**

O tempo médio desde que um problema era registado até ser concluído era de aproximadamente 9 dias, sendo verificado um pico no *sprin*t 4 em que atingiu os 74 dias. Dada a discrepância de dados, este valor foi calculado com base no cálculo da média ponderada. Através da observação da Figura 10 nota-se uma queda abrupta no tempo médio até resolução. Isto deveu-se ao facto de ter surgido a figura do "bombeiro" – elemento da equipa *tech* - que tinha como responsabilidade gerir todo o processo. Apesar de se ter reduzido o tempo, este continuava longo e todo o demais processo semelhante, persistindo os mesmos problemas e apesar de existir o "bombeiro", o reportamento de problemas continuava semelhante sendo a principal diferença o facto de este iniciar o registo, ainda que incompleto, de grande parte dos problemas reportados.

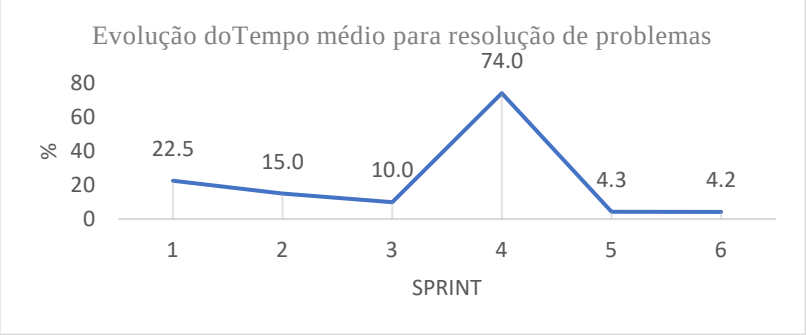


Figura 10 – Evolução do tempo médio até resolução de problemas reportados via *helpdesk*

As causas identificadas para a existência deste problema são:

- (C6.1) Não criação de conhecimento e consequente não existência de visão global

Cada vez que um problema era resolvido, a causa/problema identificado, bem como os procedimentos a seguir, não eram registados. Ao não se registar esta informação, quando um problema semelhante surgia mantinha-se inalterado o tempo de resolução, em vez de o diminuir. O não registo da causa escrita não permitia detetar a origem dos problemas, não existindo por isso conhecimento sobre o estado atual dos processos, não sendo aqui cumprido o terceiro princípio *Lean*, o de criar conhecimento. Parte desta causa acaba por ser resolvida com a identificação de problemas feita.

- (C6.2) Fluxo confuso e demasiado complexo

Pode-se começar por evidenciar o fluxo complexo e confuso de reporte de problemas como uma das causas mais impactantes, com especial foco no facto de os *stakeholders* poderem solicitar pedidos através de variadas formas sem que a informação seja registada em local único e sem que haja um procedimento a seguir, tal como exposto no capítulo 3.3.4, isto é, sem qualquer uniformização.

- (C6.3) Não priorização dos pedidos

A não priorização dos pedidos é outro fator que tem um importante impacto no mau funcionalmente deste sistema, não se seguindo nenhum método de priorização, nem qualquer outro critério para organização dos pedidos. Por fim, um dos fatores que poderá ter mais impacto neste processo poderá ser a não existência de nenhum indivíduo a quem os *stakeholders* reconheçam a responsabilidade pelo serviço de *helpdesk* e acabem por não saber a quem devem reportar os problemas, tal como Datta e Lade (2015), Akinnuwesi et al. (2014) e Siti-Nabiha, Thum e Sardana(2012) sugere que devem existir. A figura do bombeiro, apesar de ter surgido como forma de mitigar este problema, devido à sua rotatividade deixava os *stakeholders* confusos sem saber a quem reportar os seus problemas, retomando os métodos a que estavam habituados.

- (C6.4) Não existência de medidas preventivas para diminuição do número de pedidos.

Para evitar o aparecimento de problemas em *helpdesk* não era tomada qualquer tipo de medida, isto é, não se faziam ações para evitar o aparecimento de problemas, o que acabava por levar ao aparecimento de problemas, em parte, semelhantes a alguns já resolvidos.

Conjuntamente, estes fatores levavam a que os problemas ficassem perdidos e esquecidos, sendo retomados, alguns deles, apenas quando a não resolução dos mesmos tinha um impacto muito grande para o negócio. Por todos estes motivos o tempo até à resolução tornava-se longo.

- **(P7) Equipas insatisfeitas**

A análise do inquérito (anexo C) permitiu identificar a avaliação e o nível de satisfação dos *stakeholders* (utilizadores) e da equipa *tech* (responsáveis pela resolução). No que toca à avaliação da qualidade de funcionamento deste sistema a maioria classificou-o como "Razoável"[10] e no que diz respeito ao nível de satisfação conclui-se que cerca de 67% da equipa estava pouco ou nada satisfeita[11] com o sistema de *helpdesk*, tal como mostra a Figura 11.

[10] A avaliação da qualidade do serviço de helpdesk estava dividida em 5 níveis: Muito mau, Mau, Razoável, Bom e Muito Bom.

[11] A classificação do nível de satisfação estava dividida em 5 níveis: Nada Satisfeito, Pouco Satisfeito, Moderadamente Satisfeito, Muito Satisfeito, Extremamente Satisfeito.

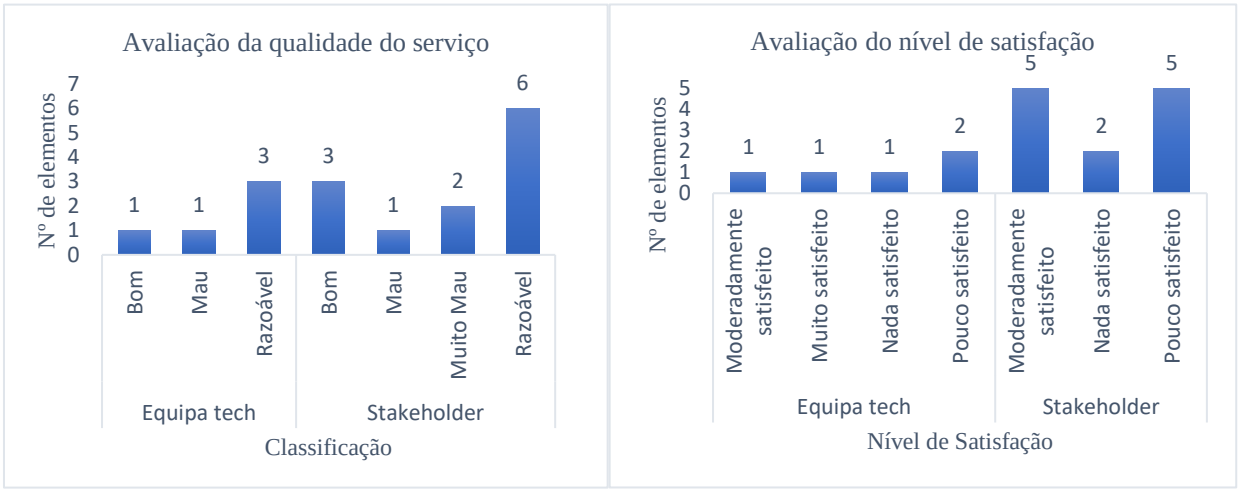


Figura 11 - Taxa de classificação e satisfação do sistema

Causas identificadas:

- (C7.1) Interrupções

No caso da equipa *tech* a existência de um elevado número de pedidos impactava as suas tarefas diárias devido à sobrecarga e sobreposição de tarefas com que ficavam. As interrupções constantes e o tempo gasto com o serviço de *helpdesk*, que os afastava e dificultava a reconcentração nas suas funções *core*, tal como Brian Bailey e Josep Konstan (2006) indicam no seu estudo, viradas no momento para o desenvolvimento da nova versão da plataforma, tinham também um elevado impacto.

- (C7.2) Tempo de espera longo e falta de feedback

No caso dos *stakeholders*, a existência de um elevado número de *bugs* e outros problemas bem como a longa espera para os resolver, afetam o trabalho dos mesmo de uma forma severa. Estes estavam ainda insatisfeitos pela falta de *feedback* que recebiam sobre: estado do problema, estimativa de tempo até resolução, problema e causa detetado, e a espera longa como já mencionado.

- **(P8) Interface do cliente sem manutenção**

Tal como referido no capítulo 3.2 e evidenciado no capítulo 4.1, a interface dos clientes deveria ser mais utilizada, advindo o motivo da baixa manutenção que se faz da mesma. O elevado número de bugs e complexidade da interface levou a que os clientes usassem o seu contacto direto com os AMs, para que estes realizassem a maioria dos procedimentos na plataforma. Este problema causou por consequência outros, como o afastamento dos AMs das suas funções *core*, a não manutenção da interface e a não existência de um serviço de apoio ao cliente centralizado.

5 Identificação e Desenho de soluções

O presente capítulo tem como objetivo a identificação de soluções para os problemas identificados no capítulo 4 em cada uma das fases do processo de desenvolvimento de *software*, (especificação, desenvolvimento e manutenção), dando assim resposta ao objetivo O2. Esta identificação tem por base a aplicação dos conceitos introduzidos no estado de arte (cap.2) para melhoria e/ou eliminação de problemas nos processos.

Para cada fase do processo apresentam-se os problemas (P) seguidos do conjunto de ações propostas (A). Estas foram desenvolvidas visando a possibilidade de sempre que possível, quantificar os efeitos de implementação das mesmas. As medidas implementadas foram: a uniformização de processos através da criação de *frameworks*; alterações na forma de funcionamento de processos e, em alguns casos, a sua restruturação total; criação de ferramentas de gestão visual; utilização de métricas de controlo adequadas; aplicação dos 5S, aplicação dos 7 princípios *Lean*, entre outros.

Por fim, no fim do capítulo, Figura 13, apresenta-se a nova matriz de responsabilidades de acordo com as alterações feitas nos processos.

5.1 Processo de Especificação

Para mitigar de forma conjunta os dois problemas identificados no capítulo 4.1:

- (P1) Necessidades dos utilizadores diferentes das funcionalidades desenvolvidas
- (P2) Não utilização simultânea da especificação escrita e visual bem como defeitos e incoerências entre os mesmos.

Propôs-se uma solução que permitisse uniformizar este processo, pelas inúmeras vantagens que traz (ver capítulo 2), dado que os problemas associados ao processo de especificação têm um grande impacto sobre as restantes fases do processo.

(A1) Uniformização do processo de especificação escrito e visual:

A uniformização do processo surge através da criação de uma *framework* (ver anexo D) para especificação de requisitos como forma de agilizar o mesmo, aumentar as taxas de execução do *sprint* (por diminuição de retrabalho), diminuir os pedidos de manutenção (por serem criadas as funcionalidades necessárias à realização das tarefas) e permitirá ainda que sejam preenchidos por completo todos os campos essenciais para a criação de uma funcionalidades evitando desvios da mesma por lapso na exposição de requisitos, dando respostas à causa C1.2 e C3.6.

A *framework* sugerida é composta por duas secções, uma para a definição da *user story* e a segunda para as tarefas necessárias para completar a mesma. A primeira pressupõe o preenchimento de campos relativos à prioridade, dificuldade, *owner* e *release manager*, valor para o negócio, utilizadores a que se destina, *flowcharts*, dependências com outras *user stories* e integração da especificação visual, dando resposta a C2.1 (no anexo D) explica-se o conteúdo de cada um destes campos). A segunda secção destina-se à apresentação de cada uma das tarefas associadas à *user story* devendo para cada uma especificar-se os requisitos de especificação (ver anexo D). Esta medida ganha maior relevância ao servir de apoio à implementação de um departamento para assegurar a qualidade, que terá por objetivo verificar o especificado e compará-lo com o produzido antes de ser *released* (ver capítulo 5.3).

Verificou-se que esta solução não resolvia por completo o problema P1, propondo-se como forma de complementar a medida atrás mencionada que se passa a descrever.

(A2) Modificação do processo de especificação:

Como forma de direcionar a plataforma ainda mais para o cliente propôs-se aumentar o diálogo com o mesmo, o que está de acordo com o preconizada por Hibbs, Jewett e Sulivan (2009) e

Mary e Tom Poppendieck (2003) bem como do pressuposto pelas metodologias *Agile*, dando resposta a C1.1. Para isso, após o normal processo de especificação de requisitos, adicionou-se mais um passo no processo de especificação exposto no capítulo 3.3.2. Após a realização do normal procedimento antes de se entregar os requisitos à equipa *tech* para produção sugeriu-se a introdução de uma segunda reunião com o cliente. Assim, a equipa de produto deverá reunir-se com o cliente em pelo menos 2 momentos, um primeiro para o levantamento geral dos requisitos e um segundo para retificação e confrontação entre o pedido inicial e o realmente desejado. A estrutura desta reunião segue uma lógica diferente do primeiro contato com o cliente, sendo que o objetivo desta deverá passar pela apresentação de um protótipo da solução proposta. Assim e através da elaboração de um conjunto de tarefas, a serem realizadas pelo cliente no referido protótipo, pretende-se obter *feedback* deste já a nível funcional e não tanto numa vertente conceptual, para que não se envie para produção funcionalidades não aprovadas. Dada a impossibilidade de falar com todos os clientes, escolheram-se clientes chave, clientes com grande valor para a empresa e, simultaneamente, com negócios diferentes (ex.: *webshops*, retalhos, *webshops* e retalhos).

Estas medidas têm também como objetivo tornar a plataforma mais amigável para os seus utilizadores.

5.2 Processo de Desenvolvimento

Nesta fase foram identificados, no capítulo 4.2, dois grandes problemas:

- (P3) Taxas execução de *sprints* irregulares e tendencialmente baixas;
- (P4) *Daily meetings* longas e sem cumprimento do seu objetivo.

No que diz respeito às **taxas de execução de *sprint* baixas (P3)** e como forma de tornar quantificáveis os problemas encontrados no capítulo 4.2 bem como as medidas tomadas, numa primeira fase introduziram-se as seguintes métricas de controlo:

- Taxa de concretização;
- Rácio entre tarefas resolvidas e *backlog*;
- Imprevistos por *sprint.*

Como forma de mitigar os problemas identificados implementaram-se medidas tendo por base a aplicação de conceitos *Lean*.

(A3) Implementação de gestão visual e adaptação do processo ao mesmo

A gestão visual sugerida e implementada foi pensada para que simultaneamente desse resposta às baixas taxas de execução de *sprints* (P3) e às longas durações das *daily meetings* (P4).

Para mitigar a baixa visibilidade sobre o que estava a ser discutido, projetou-se um quadro de grandes dimensões no local onde se realizavam as *daily meetings*, Figura 12, com o objetivo de facilitar a discussão diária. A estrutura do quadro foi desenhado a pensar na possibilidade de se visualizar um *sprint* completo, com o nível de detalhe ao dia, procedendo-se à sua implementação, dando assim resposta a C4.1. Este, tal como mencionado atrás, não foi construído unicamente com este objetivo, mas também para permitir um rápido mapeamento da dependência entre tarefas, como forma de dar apoio às medidas a seguir mencionadas, relativas ao aumento da taxa de execução, dando resposta C3.3.

Este quadro posteriormente sofreu uma iteração visando atingir ainda mais objetivos, tais como permitir às equipas percecionarem o estado de desenvolvimento de cada *feature* e gerir o processo de teste e gestão de *releases*. Para facilitar a discussão diária a estrutura do quadro possibilitava a gestão ao dia, permitindo a visão de um *sprint* completo, já como acontecia com a versão anterior.

Para complementar, e como forma de permitir uma melhor gestão do *sprint* implementou-se um outro quadro em *high level* para gestão ao épico e *user story* e não à tarefa, visando assim assemelhar-se a um *roadmap* a curto prazo, com o objetivo de transmitir à equipa e aos stakeholders internos o planeamento a 1/2 meses.

Como forma de dar mais visibilidade aos diferentes tipos de objetos nos quadros atribuiu-se a cada um uma cor diferente. A estrutura para cada cartão também foi definida como tentativa de agilizar o processo e facilitar a leitura.

Na segunda interação do quadro eliminou-se o uso dos *post-its*, criando-se cartões reutilizáveis, já com a estrutura pré-definida nos mesmo (ver anexo E), com o objetivo de não só economizar e ser ecológico, mas também de diminuir o tempo de preparação, por *sprint*, do quadro, permitindo a qualquer elemento preencher os cartões sem ter de estar a confrontar a *framework*, uma vez que passava a ser apenas necessário o preenchimento dos espaços.

Para que o quadro cumprisse o seu objetivo o processo de colocação dos cartões, em ambas as versões, foi uniformizado. Após a realização do *sprint planning* a equipa de produto deve verificar as tarefas que ficaram alocadas para o *sprint* bem como o respetivo elemento responsável, criando um cartão para cada tarefa. De seguida, a equipa *tech* deve alocá-los no dia em que planei dar como determinada a referida tarefa. Assim estariam aqui previstos o cumprimento dos objetivos de aumento de visibilidade das tarefas a realizar para o *sprint* e dependência entre estas, dando assim resposta a C3.3. Com esta medida pretende-se atingir as vantagens preconizadas por Imai (2012) (gestão visual) e Dennis (2002) (uniformização do processo).

Figura 12 - Primeira versão do quadro das daily meetings

(A4) Alteração da duração dos *sprints*:

Para determinar o período mais favorável de duração de cada *sprint* e de acordo com os limites estabelecidos por Schwaber e Sutherland (2013), pretendeu-se submeter a equipa a *sprints* com diferentes durações de 1, 3 e 4 semanas para confrontação com os *sprints* de 2 semanas. A entrada de um novo cliente, na empresa, levou à necessidade de restruturação de algumas funcionalidades. Dada a escassez de tempo e elevada urgência dessas alterações criou-se a necessidade de diminuir os *sprints*, de duas para uma semana, como forma de fomentar um controlo mais apertado. Esta alteração foi oportuna para o presente projeto pois permitiu gerir todo o processo de desenvolvimento, mas agora para *sprints* mais curtos e desta forma tentar analisar e compreender as diferenças de comportamento da equipa perante durações de *sprint* diferentes, o que permitiu em parte dar resposta a C3.2.

(A5) Alterações no processo de planeamento e no cálculo da taxa de execução:

Para evitar que a equipa seja prejudicada com o aparecimento de imprevistos aconselha-se que, ao tempo total disponível de um *sprint* se retire 10% do planeado, como forma de prevenção. Para *sprints* de 2 semanas considerando oito dias úteis de trabalho, os 10% correspondem a um dia de trabalho. Este dia, em que se considera a equipa como não disponível, poderá ser "usado"

para evitar que o possível aparecimento de imprevistos, tal como pedidos de manutenção, impeça a equipa de completar o *sprint*, dando resposta a C3.5.

Para que a taxa de execução fosse o mais completa possível e tendo já em conta as medidas tomadas no capítulo 4.2 (cálculo da taxa), verificou-se que seria insuficiente e desajustado calcular uma taxa unicamente baseada no número de tarefas resolvidas *versus* número de tarefas alocadas ao *sprint*. A solução proposta reside no cálculo da velocidade de execução da equipa tendo em conta o número de imprevistos e consequente diminuição do tempo disponível, sendo criado um *Excel* em que basta colar a informação retirada do *JIRA* e indicar o número de dias perdidos com imprevistos, sendo filtrados os pedidos de *helpdesk* e calculada a taxa de execução (anexo F). Esta medida conjunta tem como objetivo dar resposta a C3.1 e C3.5, mitigando o efeito de imprevistos, tal como estudado por Bailey e Konstan (2006).

(A6) Outras medidas:

Para tornar a equipa *cross-functional* e caminhar nesse sentido, como sugerido por Steffen (2011), recomenda-se que, em alturas menos críticas de produção, lhes sejam atribuídas tarefas de *sprint* ou de *helpdesk* (manutenção) que forcem à saída da zona de conforto e permitam que cada elemento desenvolva mais competências conseguindo assegurar o cumprimento do trabalho completo ao fim de cada *sprint*, sem que o problema, para a não completação do *sprint*, seja a falta de recursos com capacidade, respondendo a C3.8.

(A7) Redução do número de tarefas em *backlog*

Para evitar o acumular de tarefas e que a equipa sinta que por mais que avance existe sempre um enorme conjunto de tarefas para realizar, aconselha-se que o planeamento seja mais regrado e passado à equipa de desenvolvimento a curto / médio prazo em vez de a longo prazo, dando resposta a C3.7.

No caso das ***daily meetings*** (P4) o processo foi semelhante iniciando-se pela introdução das seguintes métricas de controlo:

- Tempo de duração da reunião,
- Número de presentes na reunião;
- Número de intervenientes na reunião.

(A8) Uniformização da reunião:

De seguida implementou-se o esquema proposto por Schwaber e Sutherland (2013), para as *daily meetings* (ver capítulo 2.1.2), dando resposta a C4.2 e C4.3. A implementação desta estrutura visava também a eliminação de algumas práticas realizadas na reunião que fugiam do esboço da mesma e que afetavam a partilha de informações, tal como a discussão de problemas, que foi retirada para o pós reunião e a presença de *stakeholders*, que foi apenas permitida com a condição de não interferirem na reunião e levantarem as suas questões apenas após o término desta.

A mitigação dos problemas P3 e P4 passa, em grande parte, também pelas soluções propostas no capítulo 5.1.

Dada a não existência de ferramentas de gestão de *releases* para gerir o processo, mapeou-se o mesmo criando-se uma *framework* completa para criação de *demos* e *FAQ*. Este processo será discutido no capítulo 5.3, uma vez que foi construído com o objetivo primeiro de mitigar a quantidade de pedidos de *helpdesk.*

5.3 Processo de Manutenção

O processo de manutenção é um dos mais importantes aquando da utilização de plataformas digitais para o desenvolvimento e sustentação de um negócio. Sem ele, as plataformas ficam

carregadas de *bugs* e problemas processuais acabando por se tornar obsoletas. Tal como identificado por Ashcraft (2015) o objetivo de um sistema de *helpdesk* é o de eliminar as perdas de produtividade das equipas, devido ao mau funcionamento das plataformas, pretendendo-se neste caso que este sistema sirva para controlar todo o processo de manutenção, da identificação de erros à entrega de soluções. Para isso, e com o objetivo de colmatar os problemas identificados no capítulo 4.3, são eles:

- (P5) Informação dispersa e pouco visível;
- (P6) Não criação de conhecimento e consequente não existência de visão global;
- (P7) Tempo de resolução de problemas muito elevado;
- (P8) Equipas insatisfeitas;

Identificou-se a restruturação completa do processo de manutenção como a principal solução para resolver simultaneamente todos os problemas associados a este processo.

Numa primeira fase, e como forma de os tornar quantificáveis bem como às melhorias obtidas, introduziram-se seguintes métricas de controlo, parte delas com base no estudo de Akinnuwesi et al. (2014), algumas já utilizadas na fase de identificação de problemas (ver capítulo 4.3):

- Tempo até resolução ou, como conhecido na indústria, *Mean Time To Repair* (MTTR);
- Número de *tickets* de cada tipo de problema;
- Número médio de *tickets* por elemento da equipa tech;
- Número médio de *tickets* por equipa de *stakeholders/sprint*;
- Nível de serviço do sistema de *helpdesk*;
- Número de *tickets* pendentes.

O conjunto de soluções de seguida apresentadas tiveram por base a aplicação de conceitos *Lean* como os *5S*, sistemas *pull* (*Kanban*), gestão visual, os princípios *Lean* aplicados ao desenvolvimento de *software* e ainda conceitos em torno de IT *service management*.

(A9) Alteração do processo ponta a ponta:

Primeiramente introduziu-se a obrigatoriedade de reportamento de problemas através da plataforma *Trello* (aplicação dos S *Seiri* e *Seiton*), não sendo estes aceites a partir de mais nenhuma fonte, dando resposta a C5.1 e C6.2. Esta medida teve como objetivo centralizar o serviço tal como Siti Nabiha, Thum e Sardana (2012) definem que este deve ser, para que se aumentasse a visibilidade sobre os problemas e facilitasse o processo de gestão dos mesmos, através da utilização em exclusividade desta plataforma para a comunicação entre a equipa de produto e os *stakeholders* e, deste modo, não se perder nenhuma informação. Para a construção da interface do *Trello* limpou-se a que existia e construiu-se uma mais intuitiva e direcionada ao objetivo (aplicação do S *Seiso*) eliminando-se o lixo que esta possuía.

De seguida a responsabilidade pela análise e priorização dos problemas passou totalmente para a equipa de produto, dando resposta a C6.3. Identificou-se esta solução dado que, de acordo com Siti Nabiha, Thum e Sardana (2012), Akinnuwesi (2014) e ainda Data e Lade (2015), os responsáveis por este sistema devem ter um forte conhecimento sobre o negócio sendo, neste caso em concreto, a equipa de produto que reunia essas condições. Uma outra hipótese poderia ser a alocação em exclusividade de recursos para este fim, tal como Siti Nabiha, Thum e Sardana (2012) referem, mas essa solução foi posta de parte dada a indisponibilidade para a contratação de recursos apenas para esta função.

Paralelamente a esta análise a equipa de produto deveria duplicar os *tickets* do *Trello* (plataforma de contacto entre *stakeholders* e equipa de produto) para o *JIRA* (plataforma de gestão dos *sprints*) como forma de, aqui também, aumentar a visibilidade dos pedidos de *helpdesk* para ambas as equipas e permitir um rastreamento dos mesmos. Após a resolução dos problemas cabe à equipa de produto comunicar aos *stakeholders* toda a informação que

considerar relevante, dando resposta a C7.2. Para que se entenda a escolha do *Trello* como a plataforma a ser utilizada, bem como o funcionamento do processo na mesma é, de seguida, feita uma descrição detalhada documentando-se o fluxo de um *ticket.*

O *Trello* foi a plataforma escolhida devido às suas funcionalidades uma vez que permite a criação de colunas para as quais se podem arrastar os *tickets* (anexo G). Assim criaram-se 7 colunas, uma primeira para colocar os cartões de procedimento a segunda *Backlog* para onde os *stakeholders* internos deveriam arrastar os seus cartões. Quando os cartões se encontram neste local, a equipa de produto assume que estão prontos a analisar. Quando os começa a analisar deverá arrastá-los para a coluna *Analyzing*. Neste momento a equipa recebe uma notificação num canal no *Slack* para este fim, tendo sido feita a integração destas duas plataformas. De seguida quando termina de analisar o problema e passa todas as informações necessárias para a equipa *tech* através do *JIRA* deve passar o *card* para a coluna de *Analyzed*. Dada a natureza do *JIRA* quando a equipa *tech* muda o estado da tarefa, por exemplo de "To Do "para "*In Progress"* a equipa de produto é notificada via email. Neste momento poderá, no *Trello*, mudar o estado do ticket, que se encontrava em *Analyzed*, para o estado de *Working in Progress*. Quando a equipa de produto recebe, via email que o problema foi resolvido, deverá arrastar o mesmo para a coluna de *Done*. Os *stakeholders* poderão acompanhar o processo direitamente via *Trello* ou através das notificações que vão recebendo no *Slack*.

(A10) Retirar peso do processo da equipa tech, diminuindo o número de interrupções:

Para diminuir o impacto do processo para a equipa *tech* dado que, de acordo com o estudo de Bailey e Konstan (2006), o efeito de uma interrupção tem um elevado custo, agendou-se uma reunião diária às 12h, com o líder da equipa, com o objetivo de discutir cada um dos pedidos de *helpdesk*, avaliar o tempo da resolução do problema e dificuldade até fazer a sua alocação, de acordo com a disponibilidade dos membros da equipa e áreas de conhecimento e assim evitar-se a constante interrupção não planeada, como forma de dar resposta a C6.3 e C7.1.

Para melhorar o funcionamento e aumentar a visibilidade sugeriu-se ainda que os pedidos de *helpdesk* funcionassem com base em *Kanban*, como referido por (Wysk, 2018), devido à natureza e o aparecimento inesperado dos mesmos. Por este motivo, estes deixam de estar incluídos no *sprint* e, tal como discutido no capítulo 5.2, esta medida seria complementar à medida de melhoramento do planeamento através da diminuição à equipa de 10% do tempo do *sprint*, para que este seja dedicado à manutenção da plataforma. É essencial que os *sprints* sejam planeados contemplando a retirada deste tempo com destino à manutenção da plataforma.
Apesar de este processo funcionar em *Kanban*, para aumentar a visibilidade sobre estes pedidos, incluíram-se no quadro implementado para melhorar o processo de desenvolvimento (ver capítulo 5.2).

(A11) Medidas preventivas para erros de processo

As medidas preventivas identificadas foram a introdução de *demos* e de *faqs*, tal como sugerido por Songsangyos, Niyomkha e Tumthon (2012) e Poppendieck e Poppendieck (2003, 2007). Esta medida visou diminuir o número de erros de processo reportados, com enfoque nos sub classificados como de "Não conhecimento", sendo o objetivo da mesma permitir transmitir, aos interessados, a forma correta de os realizar e gerar um acesso rápido, dando resposta a C6.4.

Para ser possível introduzir estas medidas foi necessária analisar a origem dos mesmos, o que revelou que a equipa de *accounts* era a que com mais erros se deparava e por isso a mais impactada por este serviço. Assim a realização das *demos* iniciou-se pelas que mais interessariam a esta equipa.

O processo para identificação destes tipos de erros mais comuns, para de seguida se realizarem as *demos* e as *faqs*, foi iniciado com a análise individual de cada um deles passando-se de seguida para a realização de reuniões, com as equipas utilizadoras da plataforma, para a identificação dos processos em que se sentiam menos à vontade para realizar, para

posteriormente se confrontar e complementar com os já identificados. Após este processo montou-se uma página onde se colocaram as dúvidas mais comuns com que as equipas se deparavam (*faqs*) e, para cada uma delas, sugeriu-se a criação de *demos.* Para facilitar a sua utilização, a página inicial permitia o acesso ao conjunto de *faqs*. Ao clicar-se numa *faq* era-se redirecionado para uma página com todas as informações importantes sobre a funcionalidade em questão. Como forma de facilitar isto repartiram-se as *demos* em fases do processo e, para cada uma delas, registaram-se as informações mais importantes num formato texto, com a utilização de *bullet points* como sugerido por Mestre (2012) , utilização de vídeo como sugerido por Van Der Meij e Van Der Meij (2014) e Lloyd e Robertson (2012). Foi utlizada esta solução conjunto dado que Van Der Meij e Van Der Meij (2014), apesar de considerarem o uso de vídeo mais apropriado, considerarem que o uso de vídeo e texto também é eficiente (ver anexo H). Esta medida foi implementada com o apoio da equipa de *account managers*, para a criação de *demos* sobre os processos que estes realizam e com o apoio da equipa de operações.

(A12) Medidas preventivas para erros Tecnológicos

Como forma de diminuir o número de problemas "Tecnológicos" sub classificados como "*Bugs*" organizou-se o procedimento de realização de testes para assegurar a qualidade e o cumprimento dos requisitos na construção das funcionalidades antes do seu lançamento, aplicando-se princípios *Lean,* com enfoque no de Integração de qualidade, respondendo a C6.4.

Os defeitos, em *software,* podem ocorrer em qualquer fase do desenvolvimento e até mesmo após o lançamento de uma funcionalidade para o mercado. Quanto mais cedo se procede à sua correção menor serão os custos associados (Pandit e Tahiliani 2015). Posto isto, no fim de cada *user story* esta deve ser submetida a teste seguindo a *framework* criada, anexo I. Esta consiste na verificação do cumprimento dos critérios de aceitação, sendo apenas feita para *users storys* passíveis de se testarem em *front end*.

(A13) Recolha do tempo gasto com cada um dos problemas

Dado que o projeto apenas se iniciou no fim do *sprint* 6, iniciou-se a recolha da duração do tempo gasto a partir do *sprint* 7 e até ao 11. O tempo foi obtido com base no diálogo individual com cada um dos elementos da equipa *tech*, em que para cada um dos problemas resolvidos por este foi pedida a estimativa do tempo despendido, esta conversa ocorreu sempre no fim de cada *sprint*, como forma de dar resposta a C6.1.

(A14) Uniformização do processo de reportamento:

Após a análise feita aos tipos de problemas e como forma de o agilizar procedeu-se à uniformização (aplicação S *Seiketesu*) do processo de reportadamente, dado que e de acordo com Dennis (2012) a uniformização de processos permite eliminar alguns processos desnecessários, pretendendo-se eliminar (aplicação S *Seiri*) a necessidade de pedir ao *stakeholder* que dê mais informações do que as que já deu. Posto isto criaram-se *frameworks*, visando não só um reportamento de problemas mais rápido, com a informação necessária, mas também uma análise dos problemas eficiente. As *frameworks* foram criadas de acordo com os problemas mais comuns e criou-se ainda uma *framework* geral acessível para qualquer tipo de problema. Paralelamente, criou-se um conjunto de 9 etiquetas, 4 para priorização (nível de urgência), 2 para identificação da entidade afetada (*stakeholders* interno ou *stakeholders* externo) e 3 para definir o tipo de erro (processo, tecnológico ou outro) (anexo J), resposta a C6.2

(A15) Introdução de ferramentas de Gestão visual

Para complementar todo este processo e permitir ter a equipa *tech* sempre a par do estado de *helpdesk* bem como os *stakeholders* introduziram-se ferramentas de gestão visual, como o *Power BI* que permitiu a análise dos dados recolhidos de *helpdesk* e a exposição das métricas (anexo L).

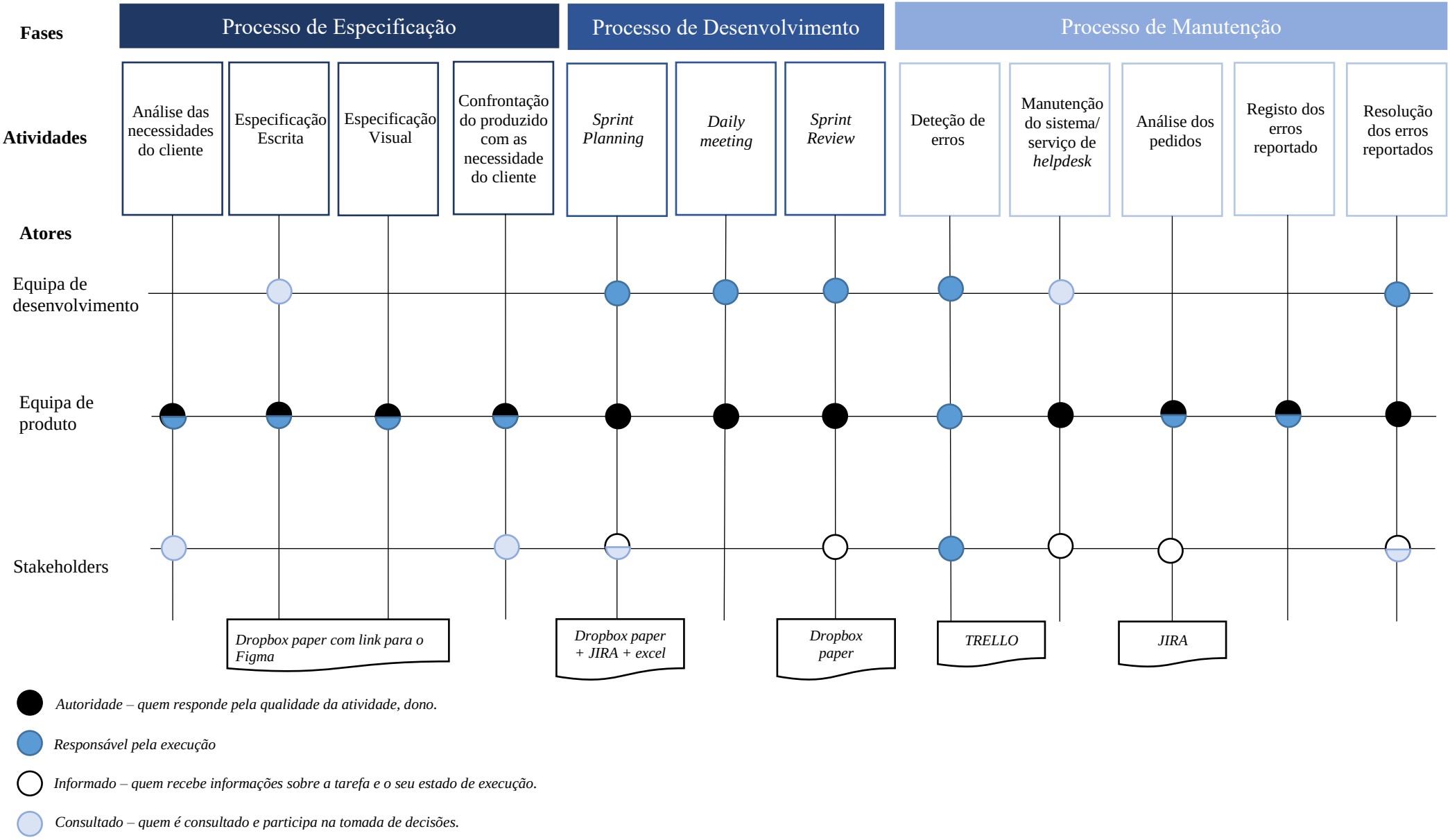


Figura 13 - Matriz RACI dos 3 processos envolvidos no desenvolvimento de Software, “TO BE”.

6 Resultados das soluções implementadas

O presente capítulo tem como objectivo apresentar os resultados obtidos através da implementação, na empresa onde o projeto foi desenvolvido, das acções propostas ao longo do capítulo 5 para cada uma das fases do processo de criação de *software*, dando resposta ao objetivo O3. Como forma de determinar quais as soluções que deveriam ser implementadas em primeiro lugar, realizou-se uma análise RICE (ver capítulo 2.2.2). Esta foi feita estimando o alcance, o impacto, confiança e esforço a curto prazo.

Desta análise resultou que o primeiro processo sobre o qual se deveria dar mais atenção seria o de Manutenção, pelo elevado alcance que tem e impacto a curto prazo, que resultou num RICE *score* elevado. Seguido pelo processo de Desenvolvimento e só depois pelo processo de Especificação, tal como se mostra na Figura 14. A longo prazo o processo de Especificação teria um RICE *score* superior dado que um processo bem feito reduz os problemas que existem na fase de desenvolvimento e na fase de manutenção.

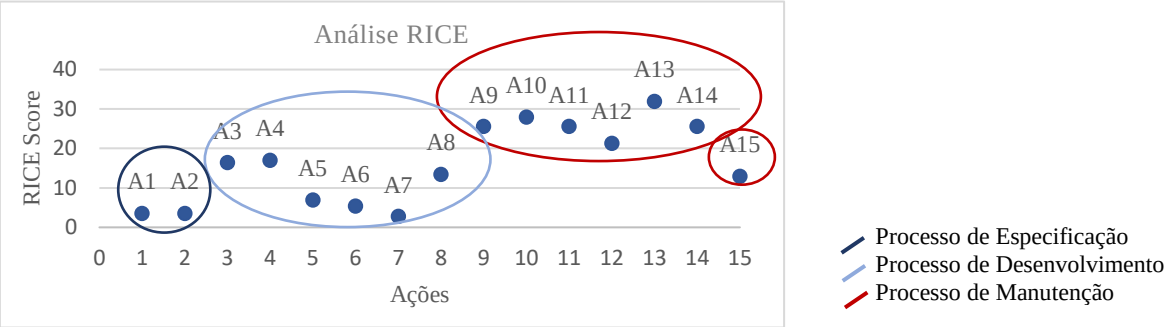


Figura 14 - Análise RICE para priorização da implementação das ações de acordo com os processos mais urgentes.

Importa notar que nem todas as medidas foram implementadas, servido o presente capítulo para analisar os resultados obtidos das medidas implementadas. A Tabela 6 serve para identificar as medidas implementadas e as não implementadas

Tabela 6 – Revisão de Problemas, Causas e Ações (implementadas e não implementadas)

	Facto (problema)	Causa	Ação	Implementada?			Facto (problema)	Causa	Ação	Implementada?	
				Sim	Não					Sim	Não
Especificação	P1	C1.1	A2	X			**P4**	C4.1	A3	X	
		C1.2	A1	X				C4.2	A8	X	
	P2	C2.1	A1	X				C4.3	A8	X	
Desenvolvimento	P3	C3.1	A5	X		Manutenção	**P5**	C5.1	A9	X	
		C3.2	A4	X			**P6**	C6.1	A13, A15	X	
		C3.3	A3	X				C6.2	A9, A14	X	
		C3.4	-		X			C6.3	A9, A10	X	
		C3.5	A5	X				C6.4	A11, A12	X	
		C3.6	A1	X			**P7**	C7.1	A10	X	
		C3.7	A7		X			C7.2	A9	X	
		C3.8	A6		X		**P8**	-	-		X

6.1 Processo de Especificação

No processo de especificação foram identificados 2 grandes problemas (P1, P2) bem com as suas respetivas causas. Como forma de tentar diminuir o impacto destas nos restantes processos identificaram-se 2 ações para os mitigar (A1, A2).

A solução A1 carece de medição sendo o seu impacto visível no capítulo 6.2 com o aumento nas taxas de execução que, pode dizer-se em parte, aumentaram pela organização e uniformização do processo, dado que e de acordo com Dennis (2002) a uniformização permite reduzir a variabilidade do processo e eliminar etapas que não tragam valor, neste caso específico, fala-se da eliminação de retrabalho, pela criação de funcionalidades diferentes das desejadas.

A ação A2 levou à restruturação do processo de especificação. Das reuniões feitas, cinco no total, pediu-se a cada cliente a execução de tarefas num protótipo não funcional. Ao observar-se a execução destas e juntamente à recolha do *feedback* dos clientes, identificaram-se algumas falhas nos protótipos, tanto a nível funcional, como a nível das operações e até do nome dado a determinados botões. De seguida procedeu-se à alteração dessas falhas, na tentativa de aproximar o que seria desenvolvido às necessidades dos clientes, procedendo-se a melhorias e, em alguns casos, à restruturação completa da funcionalidade. A longo prazo espera-se que esta medida permita, em escala, aumentar a utilização da *interface* do cliente na nova versão da plataforma, diminuir o número de pedidos de *helpdesk* por existência de limitações e diminuir/eliminar o retrabalho por parte da equipa *tech* sobre as funcionalidades pela não entrega de soluções que respondam de forma completa às necessidades do cliente.

Nenhum dos resultados foi possível de quantificar sendo que seria interessante submeter metade da equipa à nova estrutura e a restante metade à estrutura antiga, para uma mesma funcionalidade e assim conseguir-se comparar os resultados. Tal não foi possível dada a urgência de desenvolver a nova versão da plataforma o mais depressa possível e esta medida ter apenas como objetivo quantificar as melhorias. Os resultados destas duas ações estão espelhados nos capítulos 6.2 e 6.3. Espera-se que a eliminação e medição destes problemas seja conseguida após o lançamento da nova plataforma pelo que é necessário continuar a desenvolvê-la, lançá-la para o mercado e, posteriormente, ir recolhendo o *feedback* dos clientes, como forma de obter resultados práticos sobre estas medidas.

6.2 Processo de Desenvolvimento

Como forma de melhorar o processo de desenvolvimento e após a identificação dos problemas P3 e P4, bem como as respetivas causas, identificaram-se 6 soluções para melhorar este processo. Destas 6 uma, a A4, foi apenas parcialmente implementada e duas não foram implementadas, a A6 e A7.

Para o problema P3 – taxas de execução tendencialmente baixas - e como forma de o resolver implementaram-se as medidas A3, parte da A4 e a A5, sendo que as ações A1 e A2 tiveram também grande influência neste processo.

A implementação de gestão visual e todo o processo implementado envolvente (A3) permitiu melhorar a visualização das tarefas abertas para o *sprint*, possibilitando visualizar os atrasos que iam acontecendo na entrega das tarefas e assim ter uma maior visibilidade sobre as interdependências, trazendo os benefícios identificados por Imai (2012) relativos à gestão visual. Esta foi também conseguida com as alterações que se fizeram às *daily meetings* apresentadas em seguida.

A medida A5 **"**Alterações no processo de planeamento e no cálculo da taxa de execução" introduziu alterações ao cálculo desta métrica, de modo a não penalizar a equipa quando ocorriam imprevistos. A nova forma de gestão do *sprint*, deixando uma margem de 10% do

tempo disponível, para resolução de *tickets* de *helpdesk* e outros imprevistos, permitiu melhorar as taxas de execução do *sprint*, uma vez que se passou a planeá-los contemplando tempo para a manutenção da atual plataforma. Assim na última quinta-feira de cada *sprint* a taxa de execução passou a ser apresentada à equipa para que os mesmos começassem a ter uma maior perceção, através da utilização desta métrica, de como correu o *sprint* para a equipa, deixando de se aumentar a carga exageradamente quando o *sprint* corria bem.

A Figura 15 apresenta as taxas de execução calculadas, desde a data de início de projeto até ao *sprint* 11. O *sprint* 7 foi apenas de observação e início de introdução de algumas medidas, não se vendo melhorias suficientes ainda. No *sprint* 10 ocorreu um imprevisto, o que impediu a equipa de trabalhar no *sprint* durante 4 dias, o que explica a baixa taxa, comparativamente às restantes.

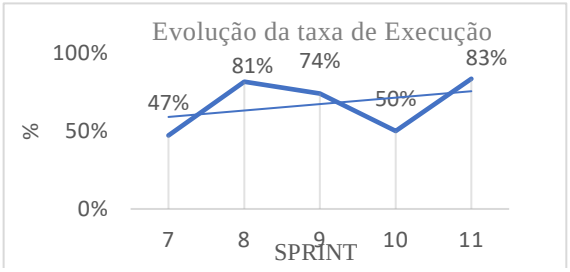


Figura 15 - Taxa de execução entre os sprints 7 e 11, para sprints de 2 semanas.

Comparando estes valores, com os anteriormente obtidos, em média a taxa aumentou 17%, isto significa que o processo de planeamento também melhorou na mesma medida, sendo apenas possível com todas as ações implementadas.

A submissão da equipa a diferentes durações de *sprints* (A4), mais concretamente a *sprints* de 1 semana, não tendo sido possível experienciar outras durações, ocorreu durante 4 *sprints*. Para isto as únicas alterações que se fizeram foram a passagem das reuniões que aconteciam ao longo de duas semanas para uma e a redução do tempo disponível para apenas 50%. Os resultados obtidos foram os que se observam na Figura 16.

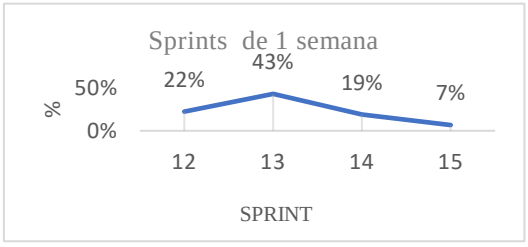


Figura 16 - Taxa de execução entre os sprints 11 e 15, para sprints de 1 semana

Com isto, pode concluir-se que tanto a equipa de produto como a equipa *tech* respondem melhor a *sprint*s de 2 semanas comparativamente aos de 1 semana. Fica a faltar a comparação com *sprints* de 3 e 4 semanas.

Com estas medidas o problema P3 foi eliminado, apesar de as taxas de execução ainda não serem as pretendidas, notou-se uma clara melhoria, que se espera que com o passar de mais alguns *sprints* e após a fase de adaptação a estas novas metodologias, estas taxas cresçam ainda mais, mantendo-se acima dos 90%.

Como forma de colmatar o problema P4 - *Daily meetings* longas e sem cumprimento do seu objetivo – tomaram-se as ações A3 e A8.

A conjugação destas duas ações permitiu diminuir o tempo de duração das reuniões. Apesar de a empresa ter partilhado os valores médios de duração das reuniões bem como todos os problemas inerentes a essa longa duração, durante um *sprint*, e antes de se alterar o processo,

decidiu-se fazer o registo diário dos valores uma vez que não tinham sido armazenadas estas informações. Na Figura 17 observa-se de forma clara a quebra na duração da reunião passando de uma média, com base no *sprint* 7, de 32 minutos para uma média de 12 minutos para os restantes, o que representa uma redução em quase 62% do tempo.

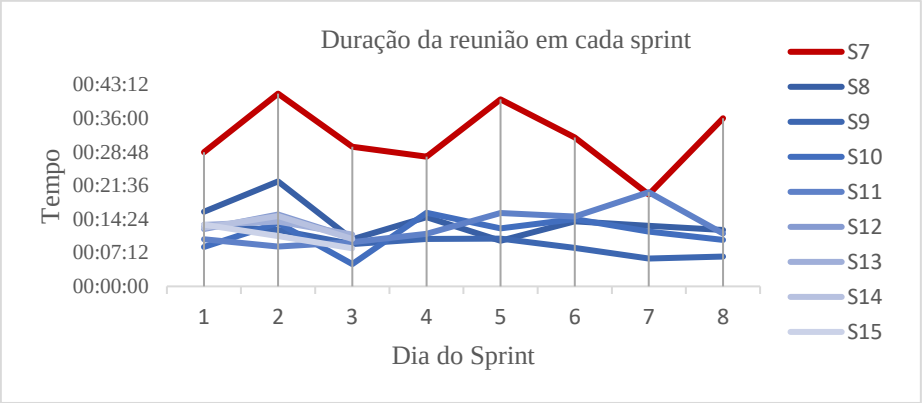


Figura 17 – Duração das daily (7-11 sprints de 2 semanas, 12-15 sprints de 1 semana)

Esta redução foi importante representando o aumento do foco dos participantes que ao seguirem a *framework*, A8, partilhavam apenas a informação pertinente em vez de abrirem espaço à discussão de outros assuntos que deveriam ser feitos noutra ocasião. A não permissão de intervenção por parte dos *stakeholders* também poderá estar na origem desta diminuição de duração.

Ao passar-se a reunião de um espaço pequeno com pouca visibilidade para um local mais amplo e com o apoio de um quadro de grandes dimensões, A3, eliminou-se a dificuldade de acompanhamento da mesma eliminando-se por isso o ruído que existia na comunicação e tornando a transmissão mais clara e fácil de acompanhar para todos. A estrutura do quadro permitiu o acompanhamento das tarefas ao dia e facilitou ainda mais a discussão, limitando-se esta apenas ao que estava a ser desenvolvido no dia.

Importa notar que a estruturação desta reunião não implicou a necessidade de criar mais reuniões, teve sim o efeito contrário, uma vez que a equipa ao seguir a *framework*, expunha toda a informação necessária apontado sempre, tal como pressuposto pela 3ª pergunta, os *blockers*, (ver cap.2.1.2) que se refletiam em dependências entre tarefas. Conclui-se assim que apesar de a duração da reunião ser bastante mais pequena, esta tinha o tempo necessário para a troca adequada de informações.

Assim verificaram-se o alcance das vantagens indicadas por Schawber e Sutherland (2013), conseguindo-se o alinhamento de toda a equipa, identificação de impedimentos no desenvolvimento de *software*, promoção de decisão rápida, promoção do aumento do nível de comunicação e ainda eliminação de outras reuniões. A resolução deste problema, P4, poderá ser indicada como um dos fatores que permitiu melhorar o problema P3.

6.3 Processo de Manutenção

A implementação das medidas propostas no capítulo 5.3 permitiram mitigar os problemas P7 e P8 de uma forma bastante boa e quantificável, conseguido em grande parte devido à resolução dos problemas P5 e P6.

O problema, P5, "Informação dispersa e pouco visível" foi eliminado na totalidade com a implementação da nova estrutura do sistema de *helpdesk* (A9). Esta medida inseriu a obrigatoriedade de registo de todos os problemas e o uso de apenas uma plataforma de contacto entre o responsável de *helpdesk*, agora a equipa de produto e os *stakeholders*; converteu-se na compactação da informação; no aumento do número de registos e numa gestão constante do

sistema, o que é visível pelo aumento do número de registos e aproximação entre os pedidos reportados e os pedidos resolvidos, tal como se observa na Figura 18. Assim conclui-se que a atribuição da responsabilidade deste processo à equipa de produto e utilização de plataforma de reportamento única, foram duas das principais alterações com mais impacto e que permitiram agilizar todo o processo, dando total visibilidade e transparência do mesmo às equipas.

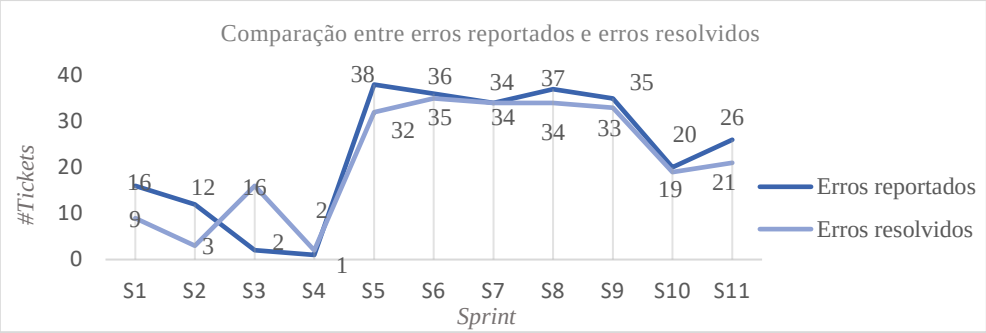


Figura 18 - Comparação entre o número de erros resolvidos e o número de erros reportados, em cada sprint

O problema, P6, "Não criação de conhecimento e consequente não existência de visão global" não foi eliminado por completo. A parte "não existência de visão global" foi resolvida logo no processo de identificação de problemas (4.3) que permitiu classificar e identificar a frequência de cada tipo de erro, obtendo-se os seguintes resultados visíveis na Tabela 7. Esta medida em conjunto com a medida de recolha do tempo despendido com a resolução de erros (A13) permitiu obter os seguintes resultados:

Tabela 7 - Resultados obtidos da avaliação dos problemas entre o sprint 1ao 11 e análise do tempo despendido por tipo de problema (em minutos)

Categoria	Sub Categoria	Frequência		T.Total gasto	T.médio/ pedido	T./ Tipologia
Tecnológico	Bugs	89%	53%	3691	87	80
(Manutenção)	Externo	11%		5	5	
Processo	Complexo manualmente	43%	33%	972	36	36
	Não conhecimento	52%		885	38	
Sugestão	-	9%		0	0	0
Limitação	-	2%		1135	82	82
Total				**7138**	60	

A eliminação de cada pedido de *helpdesk* de processo não conhecimento permite poupar 36 minutos, em média, à equipa *tech*. Estes dados permitiram concluir uma necessidade ainda maior de existência de medidas preventivas, para evitar a ocorrência de erros, com principal foco em erros de Processo de Não conhecimento, dado serem problemas que, maioritariamente podem ser mitigados com instrução do utilizador.

As medidas preventivas identificadas foram: a introdução de *demos* e de *faqs*, tal como sugerido por Songsangyos, Niyomkha e Tumthon (2012) e Poppendieck e Poppendieck (2003, 2007) e exposto no capítulo 5.3. Esta medida visou diminuir o número de erros de processo reportados e sub classificados como de "Não conhecimento" (52% dos 33% -17.16% do total), sendo objetivo da mesma permitir transmitir, aos interessados, a forma correta de os realizar e gerar um acesso rápido.

Para ser possível introduzir estas medidas foi necessário avaliar a origem dos erros e, através da análise dos dados recolhidos. A leitura da Figura 19 permite concluir que a equipa de *stakeholders* mais impactada pelos atrasos no sistema de *helpdesk* é a de *accounts*.
Assim, as primeiras *demos* construídas foram direcionadas aos problemas que mais afetavam esta equipa, passando-se a enumerar: criar clientes em *bulk* ou manualmente, criar sales *orders* em *bulk* ou manualmente, ações a tomar sobre sales *orders*, entre muitos outros processos, de acordo com os identificados como prioritárias (mais utilizados *versus* mais erros reportados).
No caso da equipa de operações após a análise dos erros e reuniões levantaram-se 4 processos em que as equipas sentiam mais dificuldade: *Stock transfer*, *Pick from stock*, *Pick* de confirmação e *Pick back to Stock*.

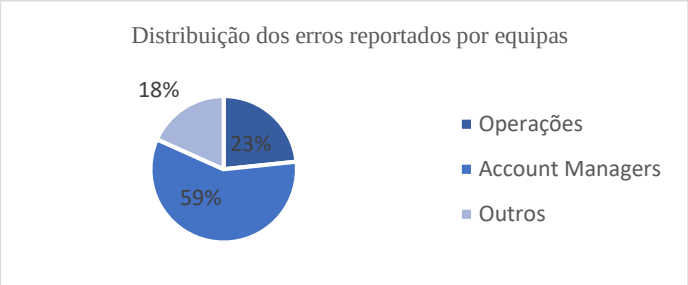


Figura 19 - Distribuição do reportamento de erros por equipas

Os resultados desta ação não foram passíveis de medição durante o tempo do projeto dado que os picos de reportamento de erros ocorrem nos picos das *seasons*. Estes só se tornariam relevantes na comparação entre picos de *season*. Dado o tempo necessário para a construção das *demos* atrás referidas, não foi possível implementá-las nos 2 meses iniciais do projeto, o que levou à perda do pico da *season* para comparação com o pico que ocorreu nas 2 últimas semanas do projeto.

No caso do departamento de qualidade, a sua implementação foi com o objetivo de reduzir o número de pedidos de *helpdesk* associados a erros tecnológicos *bugs* dado que a não existência destes erros leva a uma poupança média de 87 minutos por cada problema não reportado, tal como identificado na Tabela 7. Os resultados desta medida foram medidos devido ao lançamento de 3 funcionalidades apenas após passarem por estes testes. O número de pedidos de *helpdesk* classificados como de tecnológicos e relativos a estas funcionalidades foram zero, o que permite concluir que se todas as funcionaldades tivessem passado por este processo o número de *bugs* detetado seria consideravelmente inferior.

No que toca ao elevado tempo médio até à resolução de erros, P7, foi possível reduzir o mesmo em 70,18% passando de uma média de 9 dias para 2 dias e meio. A Figura 20 mostra a tendência de redução do tempo por *ticket*, em dias, e o aumento do registo e no lado direito observa-se o aumento do nível de serviço. É importante notar que apesar de no *sprint* 5 e 6 o nível de serviço ser bastante superior ao que aconteceu anteriormente, isto deveu-se à figura do bombeiro que ficou dedicado a dar resposta aos problemas e a partir do *sprint* 7 (adaptação) e 8 (implementação em força) deveu-se às medidas aplicadas. Note-se que apesar de o SLA (nível de serviço) nos *sprints* 9 e 10 ser de apenas 66% e 60% o tempo médio de resolução de erros foi de 3,6 e 2,1 respetivamente o que representa valores inferiores, e portanto melhores, face ao sistema de *helpdesk* anterior, em que nos *sprints* 5 e 6 foi superior a 4 dias.

O impacto destas alterações sobre as equipas foi significativo aumentando o nível de satisfação tanto da equipa *tech* como da equipa de negócio de maioritariamente uma equipa pouco ou nada satisfeita (66, (6)%) para muito satisfeita (66,(6)%). Estes resultados foram obtidos através do inquérito feito (anexo C), resolvendo assim o problema, P8, “Insatisfação das equipas”.

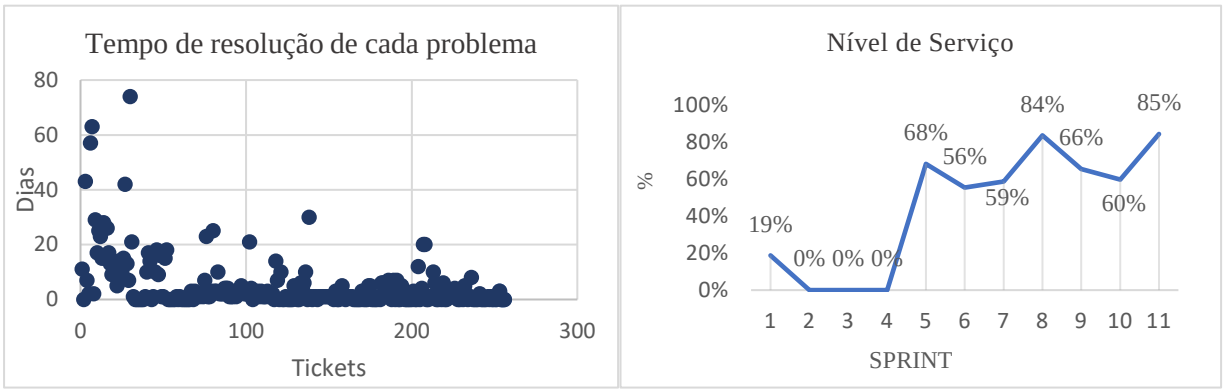


Figura 20 – Redução do tempo até resolução de problemas (esquerda) evolução do nível de serviço (SLA) calulado com base num tempo de resolução ideal de até 2 dias (direita).

A obtenção destes resultados positivos foi apenas possível com a resolução total e/ou parcial dos restantes problemas identificados nesta fase no capítulo 4.3 e a implementação da maioria das soluções propostas.

Em suma, a reformulação de todo o processo permitiu melhorias quantificáveis e visíveis para o processo de manutenção. Adicionalmente a gestão de todo ele permitiu a identificação de um conjunto de limitações da plataforma e o planeamento de *user storys* para as corrigir, maioritariamente através de desenvolvimentos simples como colocação de filtros especificos, permissões de edição em *bulk* sobre determinados tipos de informação, preenchimento obrigatório de alguns tipos de informação, entre outras, todas com o objetivo de tornar a versão a atual mais leve e desempedir a equipa *tech* para que esta se foque o mais possível na versão nova da plataforma. Adicionalmente também se fez um levantamento de todos os *tickets* de *helpdesk* não resolvidos levantado-se junto dos *stakeholders* a urgência de resolução dos mesmos e respectivas necessidadas, criando-se *user storys* nos casos em que se considerou oportuno.

6.4 Trabalhos realizados consequentes da análise dos resultados

No fim deste processo identificaram-se os seguintes problemas: equipa de produto subcarregada, duplicação de tarefas em 2 plataformas, surgimento de alguns problemas após a realização da reunião e pouca escalabilidade do serviço para o crescimento esperado. Como forma de se eliminarem estes problemas desenvolveu-se um projeto complementar de melhoria ao sistema de *helpdesk* já apresentado. Apesar de não ter sido implementado o projeto encontrava-se pronto, à data do fim da dissertação, bem como o seu *roadmap* de lançamento. Os objetivos deste projeto complementar são então: reduzir o tempo que a equipa de produto gasta com *helpdesk*, reduzir manualismos, dar mais *feedback* às equipas, permitir escalabilidade.

Para isto redesenhou-se o processo de *helpdesk* (ver anexo M) através da utilização de uma plataforma, o *Jira Service Desk* e que, através da integração com o *slaasck* e o *slack*, permite ao utilizador, na própria plataforma da empresa, solicitar assistência e ser redirecionado para o canal do *Jira Service Desk*, onde tem à sua disposição um conjunto de *frameworks* customizadas de acordo com os diferentes problemas. Aí a equipa de produto só necessita de validar a informação e atribuir diretamente a tarefa ao elemento da equipa *tech*, sem necessidade de duplicação de tarefas para outras plataformas. À medida que os problemas são resolvidos o utilizador é sempre informado, por processos automáticos, aumentando a visibilidade, e o acompanhamento relativo aos estágios de resolução do problema. Assim esta solução tem como objetivo responder a todos os problemas e melhorar ainda mais este serviços, vendo-se a aplicação do método PDCA.

7 Conclusões e perspetivas de trabalho futuro

No presente projeto estudou-se a aplicabilidade de conceitos e ferramentas *Lean*, como gestão visual, PDCA, 5S, entre outros, bem como métodos adequados para aplicar às diferentes fases que compõe o desenvolvimento de *software*.

7.1 Principais conclusões

A indústria de desenvolvimento de *software* é uma indústria extremamente complexa, sendo relativamente recente, e por isso existindo um longo caminho ainda a ser feito de forma a conseguir atingir a eficiência que a indústria tradicional já consegue atingir. Este projeto teve como objetivo o estudo dos processos de desenvolvimento de *software*, aplicado a um caso de estudo específico, em que a empresa se encontrava a desenvolver uma nova versão da sua plataforma digital enquanto mantinha a atual. A empresa em questão estava na *early stage*, onde a maturidade do produto tecnológico ainda não era devidamente robusta, sendo por tudo isto complexa a obtenção de dados ou métricas que permitissem analisar o estado inicial dos processos aquando do início do projeto, o que fez com que o período de observação tivesse de ser mais longo não permitindo uma rápida análise e identificação de melhorias. Com isto conclui-se a importância da criação e existência de ferramentas para armazenamento e recolha de dados, para que seja possível o estabelecimento de métricas de consulta rápidas e consequente identificação rápida de problemas.

Neste projeto procuraram-se aplicar conceitos, de várias áreas de conhecimento como forma de enriquecer os resultados. Com isto foi possível explorar ponta a ponta o processo de desenvolvimento de uma plataforma digital, considerando-se que o foco em apenas uma das fases, como originalmente sugerido, reduziria não só os resultados, mas também a sua qualidade. Assim, partiu-se da pergunta de partida "Como melhorar os processos de Especificação, Desenvolvimento e Manutenção do *software* da empresa?" tendo como objetivo geral "Identificar problemas, analisar e aplicar soluções para melhorar os processos de Especificação, Desenvolvimento e Manutenção de *software*". O projeto foi norteado pelos objetivos específicos, que foram respondidos de forma completa ao longo da dissertação:

- (O1) Identificar os problemas inerentes às fases de especificação, desenvolvimento e manutenção de desenvolvimento de software (capítulo 4);
- (O2) Identificar e aplicar soluções para os problemas encontrados nos processos de especificação, desenvolvimento e manutenção de desenvolvimento de software (capítulo 5);
- (O3) Verificar a existência de melhorias após a aplicação das soluções identificadas para os processos de especificação, desenvolvimento e manutenção de desenvolvimento de software (capítulo 6).

Para evitar o atrás enunciado, que o processo fosse melhorado apenas numa fase, investigaram-se as vertentes desde a simples recolha de requisitos para criar uma funcionalidade até à necessidade de manutenção da plataforma. Esta visão global permitiu estudar e compreender a interligação extremamente forte entre as diferentes etapas de desenvolvimento de uma funcionalidade e de tudo que a rodeia. Este é um dos aspetos mais complicados quando se fala no desenvolvimento de *software* , a circularidade das atividades e o enorme número de dependências entre os processos. Por esse motivo o método *Waterfall* foi descartado servindo apenas para enfatizar a evolução que existiu nesta indústria.

Assim, foi necessário separar o processo em fases, acordando-se três fases, delimitando-se e selecionando-se os processos inerentes a cada fase. Numa primeira fase iniciou-se a observação de todos os processos. A experiência de realizar os processos e em alguns casos assumir-se a responsabilidade pelo funcionamento dos mesmos permitiu aumentar a perceção e a capacidade

de os julgar, possibilitando de uma forma mais concreta a identificação de problemas, causas e ações de melhoria. Assim identificaram-se um conjunto de problemas para cada fase.

Processo de especificação:

Da análise e execução deste processo foi possível a identificação de 2 problemas inerentes à não uniformização, estruturação e desorganização do processo. Para os eliminar recorreu-se à uniformização de todo o processo e à introdução de mais um momento de diálogo com o cliente antes de as funcionalidades serem produzidas. Os frutos destas alterações não foram completamente determinados, dado o curto período de duração do projeto, mas o *feedback* das equipas envolvidas recebido relativamente ao processo de especificação foi positivo, ficando a faltar perceber se, com a alteração do processo, se as funcionalidades produzidas correspondem às necessidades dos clientes.

Processo de desenvolvimento:

No caso do processo de desenvolvimento de *software* foram feitos ajustes a práticas já pré introduzidas, mas às quais lhe faltava estrutura e uniformização, duas coisas de grande importância para, essencialmente e devido à baixa maturidade da empresa, permitir ganhar-se experiência para posteriormente permitir mais flexibilidade sem que as estruturas quebrem. Da análise dos processos e da recolha de dados identificaram-se 2 problemas, taxas de execução baixas e *daily meetings* demoradas sem cumprir a sua função. As medidas implementadas, desde a uniformização do processo à implementação da gestão visual permitiram aumentar as taxas de execução de um modo satisfatório, em média 17%, nunca descendo abaixo dos 50%, caso que aconteceu pela ocorrência de um imprevisto que tirou 4 dias de trabalho à equipa, e permitiram ainda tornar as *daily meetings* reuniões curtas e adequadas com os corretos fluxos de informação, reduzindo a sua duração em 62%.

Assim, conclui-se que a utilização de práticas *Lean*, como uniformização de processos e o uso de gestão visual têm um impacto importante sobre a utilização de métodos *Agile*, mais concretamente *Scrum*. Deste processo também se retira que o suporte dado pela equipa de produto é fundamental para o bom funcionamento dos *sprints*, devendo a mesma gerir de forma adequada a carga a dar à equipa de desenvolvimento sem nunca aumentar a carga desproporcionalmente apenas pelo alcance de bons resultados, devendo sim fazê-lo de uma forma gradual.

Estas melhorias foram apenas conseguidas devido à construção de *frameworks*, uniformização de processos, introdução de gestão visual, o que foi fundamental para a construção de procedimentos , métricas e de objetivos. Estas métricas representam um primeiro passo para que no futuro se analise a evolução das mesmas e se tomem medidas no sentido de permitir uma melhoria continua.

Processo de Manutenção:

Este processo, à data de início do projeto, encontrava-se com inúmeros problemas, desorganizado, sendo um dos grandes motivos de descontentamento das equipas. Por este motivo, e de acordo com a análise RICE feita, para curto prazo, conclui-se que era urgente e prioritário dar uma maior atenção a este processo. Assim tomaram-se de imediato um conjunto de ações corretivas que passaram pela restruturação completa do processo, pela passagem da responsabilidade para a equipa de produto, criação de *frameworks* para reportamento de problemas, análise detalha dos problemas e categorização dos mesmos, um conjunto de medidas preventivas para os tipos de erros mais comuns e o uso de ferramentas de gestão visual. O uso conjunto das medidas implementadas permitiu uma diminuição acentuada no tempo de espera até resolução de problemas bem como o aumento do nível de satisfação das equipas, principalmente do lado da equipa *tech*. O tempo de resolução era de 9 dias em média e passou para apenas 2,5 dias, o que representou uma quebra de 70% do tempo de resolução de erros e

o nível de satisfação das equipas aumentou 2 níveis, passando de pouco ou nada satisfeito para muito satisfeito.

Daqui conclui-se que um sistema de apoio ao cliente, neste caso, interno, bem organizado, com uniformização e uma estrutura conhecido por todos os envolvidos, permite dar uma resposta, neste caso, 70% mais rápida do que sem todas estas ferramentas. Sendo um processo fundamental, não só para manutenção e resolução de problemas bem como para entender o nível de utilização de funcionalidades e as limitações que a plataforma apresenta, para posteriormente serem especificados os requisitos e desenvolvidas as funcionalidades necessárias.

O presente projeto estabelece assim uma base para a identificação de problemas nos processos relacionados com o desenvolvimento de *software* bem como a identificação e implementação de um conjunto de medidas para mitigar os mesmos.

As alterações introduzidas nos processos e as ferramentas produzidas foram adotadas pela empresa e permitiram a obtenção de resultados de satisfatórios a muito bons esperando-se que no futuro se possam obter os resultado em falta, devido ao curto período do projeto face ao perído necessário para acompanhar um funcionalidade pronta a ser utilizada pelo cliente.

7.2 Oportunidades de desenvolvimento futuras

Dado o curto período para realizar o projeto verificou-se a não implementação de algumas das ações identificadas, ficando também alguns resultados por recolher. Fala-se mais concretamente nas 2 ações, para mitigar o problema das baixas taxas de execução, como tornar a equipa de desenvolvimento *cross-functional* e reduzir o *backlog*, que não foram possíveis de implementar sendo então identificadas como trabalhos futuros, podendo ajudar a melhorar ainda mais os bons resultados obtidos.

No que toca à não obtenção de resultados sobre algumas das medidas implementadas isto deveu-se essencialmente ao fato de estas soluções serem pensadas a longo prazo, sendo necessário a passagem de tempo para retorno de resultados. Fala-se da introdução de *demos* e *faqs* e das medidas implementadas no processo de especificação. Dada a impossibilidade de a equipa de desenvolvimento experienciar durações de *sprints* de 3 e 4 semanas, também se considera o planeamento destes para um trabalho futuro. Identificam-se ainda como oportunidades futuras o lançamento do novo sistema de *helpdesk*, já pronto, e com um *roadmap* de implementação completo, com base na automatização de processos e integração das plataformas de contacto da equipa de produto, responsável pelo serviço, com a equipa *tech* e *stakeholders* num só local. Posteriormente prevê-se a abertura desse canal aos clientes da empresa, para que desta forma se consiga compreender ainda melhor as necessidades dos mesmos e caminhar no sentido de entregar ao cliente não só o que ele quer mas o que ele necessita. Como forma de complementar, indicam-se como trabalhos futuros ainda a utilização de *natural language programming* (NLP) para a categorização do tipo de erros reportados em *helpdesk*, de modo a evitar problemas inerentes à avaliação humana.

Indica-se a vigilância de todo o processo, ponta a ponta, para uma continua avaliação dos resultados, de modo a identificarem-se melhorias continuas sobre os processos, e fazer a aplicação das mesmas através do uso de conceitos *Lean* e outros que se considerem oportunos no futuro.

Por fim, e mais importante, com este projeto foi possível fazer um trabalho inicial de limpeza dos processos, estabelecimento de métricas de controlo e recolha de dados. Com as ferramentas criadas procura-se no futuro , de forma passiva, continuar a absorver estes dados para ser possível a rápida identificação dos pontos do processo problemáticos para que sejam construídas ações de melhoria , tais como a criação ou melhoria de partes específicas do *software*.

Referências

Adel, Alshamrani, e Bahattab Abdullah. 2015. "A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, e Incremental/Iterative Model." *IJCSI International Journal of Computer Science Issues* 12 (1): 106–11. doi:1694-0784.

Ahmad, Muhammad Ovais, Jouni Markkula, e Markku Oivo. 2013. "Kanban in Software Development: A Systematic Literature Review." *2013 39th Euromicro Conference on Software Engineering e Advanced Applications*, 9–16. doi:10.1109/SEAA.2013.28.

Akinnuwesi, Boluwaji A, Oluwatoyin A Enikuomehin, Faith-Michael E Uzoka, Onyekachi C Onwudike, Abayomi Osamiluyi, e Benjamin S Aribisala. 2014. "Electronic Helpdesk Support System in Tertiary Institutions in Developing Countries." *International Journal of Computer e Information Technology* 03 (06): 2279–2764. www.ijcit.com.

Ashcraft, Phillip Lynn, Susan M. Cummings, Blythe G. Fogle, e Christopher D. Valdez. 2015. "AskIT Service Desk Support Value Model." Los Alamos, NM (United States). doi:10.2172/1209464.

Bailey, Brian P., e Joseph A. Konstan. 2006. "On the Need for Attention-Aware Systems: Measuring Effects of Interruption on Task Performance, Error Rate, e Affective State." *Computers in Human Behavior* 22 (4): 685–708. doi:10.1016/j.chb.2005.12.009.

Boehm, Barry. 2004. *Balancing Agility e Discipline: A Guide for the Perplexed*. doi:10.1007/978-3-540-24675-6_1.

Datta, Subhajit, e Prasanth Lade. 2015. "Will This Be Quick?: A Case Study of Bug Resolution Times Across Industrial Projects." *Proceedings of the 8th India Software Engineering Conference*, 20–29. doi:10.1145/2723742.2723744.

Estadual, Universidade, Montes Claros, Unimontes Centro, e D C C Revista Clique. 2012. "Lean Software: Eliminação de Desperdícios No Processo de Desenvolvimento." *Revista Clique* 1 (38). www.ccet.unimontes.br/revistaclique.

Flynn, William C. 2014. "Behind the Help Desk : Career , Salary e Training Expectations" 15 (Ii): 285–92.

Galin, Daniel. 2018. "Software Quality Factors (Attributes)." *Software Quality: Concepts e Practice*, 23–44. doi:10.1002/9781119134527.ch2.

Gautam, Naveen, e Nanua Singh. 2008. "Lean Product Development: Maximizing the Customer Perceived Value through Design Change (Redesign)." *International Journal of Production Economics* 114 (1): 313–32. doi:10.1016/j.ijpe.2006.12.070.

Hibbs, C., S. Jewett, e M. Sullivan. 2009. *The Art of Lean Software Development*. Edited by Mike Loukides. First Edit. O'Reilly Media, Inc.

Hines, Peter, Pauline Found, Gary Griffiths, e Richard Harrison. 2008. "Staying Lean: Thriving, Not Just Surviving." *Lean Enterprise Research Centre, Cardiff University*, 282. doi:10.1201/b10492.

Hines, Peter, e Nick Rich. 1997. "The Seven Value Stream Mapping Tools." *International Journal of Operations & Production Management* 17 (1): 46–64. doi:10.1108/01443579710157989.

Hneif, Malik, e Siew Hock Ow. 2009. "Review of Agile Methodologies in Software Development 1." *International Journal of Research e Reviews in Applied Sciences* 1 (1): 2076–2734. doi:ISSN: 2076-734X, EISSN: 2076-7366.

Järveläinen, Jonna. 2013. “IT Incidents e Business Impacts: Validating a Framework for Continuity Management in Information Systems.” *International Journal of Information Management* 33 (3): 583–90. doi:10.1016/j.ijinfomgt.2013.03.001.

Kupiainen, Eetu, Mika V. Mäntylä, e Juha Itkonen. 2015. “Using Metrics in Agile e Lean Software Development - A Systematic Literature Review of Industrial Studies.” *Information e Software Technology* 62 (1). Elsevier B.V.: 143–63. doi:10.1016/j.infsof.2015.02.005.

Lei, Howard, Farnaz Ganjeizadeh, Pradeep Kumar Jayachandran, e Pinar Ozcan. 2015. “A Statistical Analysis of the Effects of Scrum e Kanban on Software Development Projects.” *Robotics e Computer-Integrated Manufacturing* 43. Elsevier: 59–67. doi:10.1016/j.rcim.2015.12.001.

Lloyd, Steven A., e Chuck L. Robertson. 2012. “Screencast Tutorials Enhance Student Learning of Statistics.” *Teaching of Psychology* 39 (1): 67–71. doi:10.1177/0098628311430640.

Lund, James R. 2012. “Service Desk Shuffle.” *The Bottom Line* 25 (1): 23–25. doi:10.1108/08880451211229199.

Matharu, Gurpreet Singh, Anju Mishra, Harmeet Singh, e Priyanka Upadhyay. 2015. “Empirical Study of Agile Software Development Methodologies.” *ACM SIGSOFT Software Engineering Notes* 40 (1): 1–6. doi:10.1145/2693208.2693233.

McBride, Sean. 2016. "Prioritization is a perennial challenge when building a product roadmap. How do you decide what to work on first" (disponível em: https://blog.intercom.com/rice-simple-prioritization-for-product-managers/)

Meij, Hans Van Der, e Jan Van Der Meij. 2014. “A Comparison of Paper-Based e Video Tutorials for Software Learning.” *Computers e Education* 78. Elsevier Ltd: 150–59. doi:10.1016/j.compedu.2014.06.003.

Mestre, Lori S. 2012. “Student Preference for Tutorial Design: A Usability Study.” *Reference Services Review* 40 (2): 258–76. doi:10.1108/00907321211228318.

Mohammad, Nabil, Ali Munassar, e A Govardhan. 2010. “A Comparison Between Five Models Of Software Engineering.” *International Journal of Computer Science Issues* 7 (5): 94–101. doi:10.1.1.402.9250.

Nakamura, Takafumi, e Kyoich Kijima. 2011. “Total System Intervention for System Failures e Its Application to Information e Communication Technology Systems.” *Systems Research e Behavioral Science* 28 (5): 553–66. doi:10.1002/sres.1114.

Ohno, T. 1988. *Toyota Production System: Beyond Large-Scale Production*. Taylor & Francis. https://books.google.pt/books?id=7_-67SshOy8C.

Pandit, Pallavi, e Swati Tahiliani. 2015. “AgileUAT: A Framework for User Acceptance Testing Based on User Stories e Acceptance Criteria.” *International Journal of Computer Applications* 120 (10): 975–8887. doi:http://dx.doi.org/10.5120/21262-3533.

Poppendieck, M., e T. Poppendieck. 2003. *Lean Software Development: An Agile Toolkit*. *Addilson-Wesley*. doi:10.1109/MC.2003.1220585.

Poppendieck, M., e T. Poppendieck. 2007. *Implementing Lean Software Development: From Concept to Cash*. A Kent Beck Signature Book. Addison-Wesley. https://books.google.pt/books?id=3TM0AgAAQBAJ.

Poppendieck, M. 2012. “Lean Software Development.” *IEEE Software* 29 (5): 165–66. doi:10.1109/ICSECOMPANION.2007.46.

Royce, Dr. Winston W. 1970. “Managing the Development of Large Software Systems.” *Ieee Wescon*, no. August: 1–9.

Schwaber, Ken, e Jeff Sutherland. 2013. “The Scrum Guide.” *Scrum.Org e ScrumInc*, no. July: 17. doi:10.1053/j.jrn.2009.08.012.

Shaydulin, Ruslan, e Justin Sybrandt. 2017. “To Agile, or Not to Agile: A Comparison of Software Development Methodologies,” 1–11. http://arxiv.org/abs/1704.07469.

Silva, Vanessa B S, Fernando Schramm, e Adriana C Damasceno. 2016. “A Multicriteria Approach for Selection of Agile Methodologies in Software Development Projects.” *IEEE International Conference on Systems, Man, e Cybernetics*, 2056–60. doi:10.1109/SMC.2016.7844542.

Siti-Nabiha, A.K., W.Y. Thum, e G.D. Sardana. 2012. “A Case Study of Service Desk’s Performance Measurement System.” *International Journal of Commerce e Management* 22 (2): 103–18. doi:10.1108/10569211211239412.

Smith, Aimie . (2017). "Make lean change management possible with Jira Service Desk" (disponível em: https://www.atlassian.com/it-unplugged/itsm/make-lean-change-mgmt-possible）

Sokovic, M, D Pavletic, e K Kern Pipan. 2010. “Quality Improvement Methodologies – PDCA Cycle, RADAR Matrix, DMAIC e DFSS Industrial Management e Organisation Industrial Management e Organisation.” *Journal of Achievements in Materials e Manufacturing Engineering* 43 (1): 476–83.

Songsangyos, Pradit, Wararat Niyomkha, e Suwut Tumthong. 2012. “Helpdesk Expert” 5 (4): 137–43.

Steffen, Juliana .(2011). "O mundo depende de Software" . (disponível em: https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/lean_para_desenvolvimento_de_sw_o_que__c3_a9_isso_afinal12?lang=en

VersioneOne. (2018). "12th Annual State of Agile Report" (disponível em http:// stateofagile.versionone.com/).

Vijayasarathy, Leo, e Charles Butler. 2015. “Choice of Software Development Methodologies - Do Project, Team e Organizational Characteristics Matter?” *IEEE Software* PP (99): 1–1. doi:10.1109/MS.2015.26.

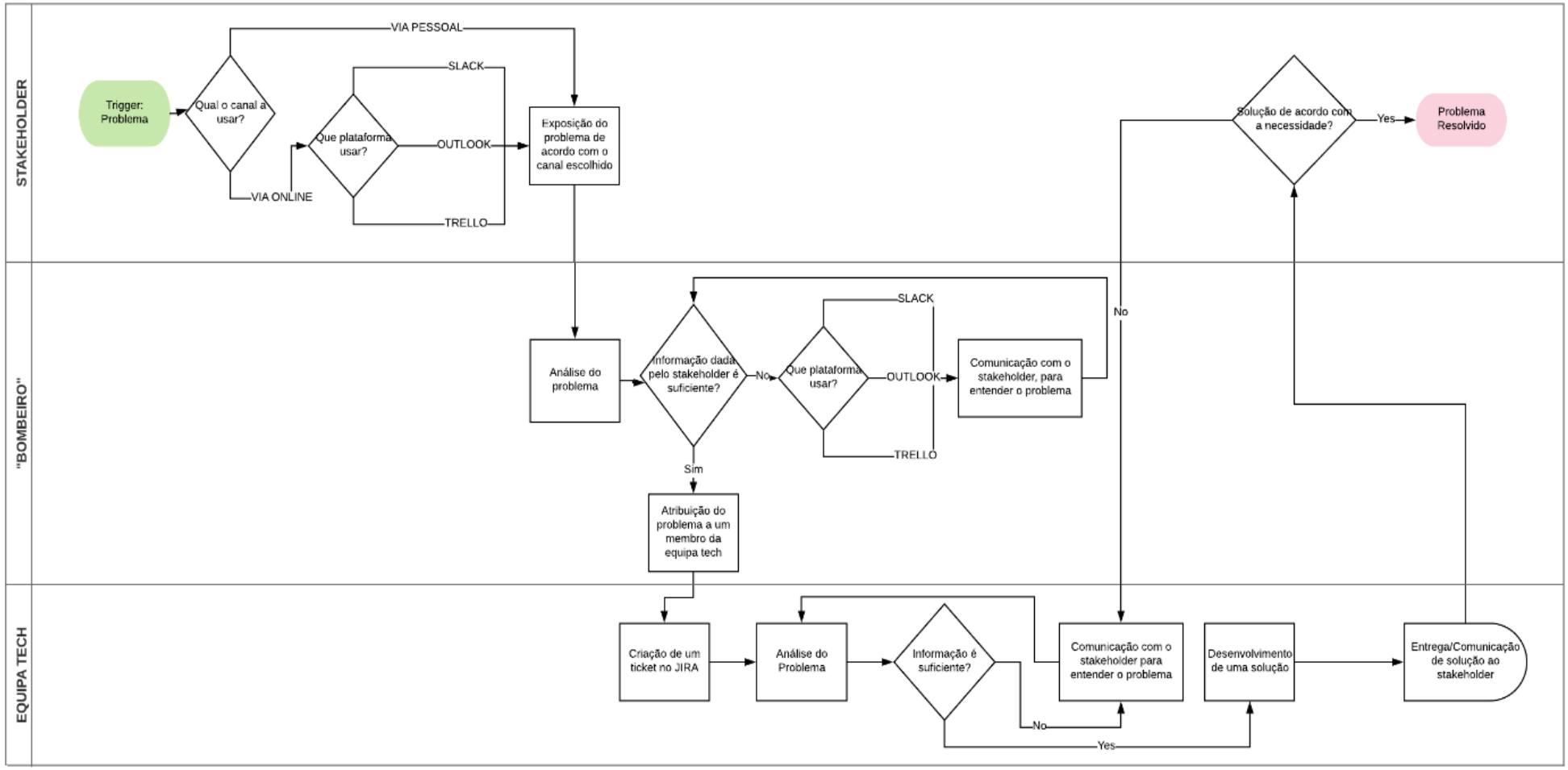
Womack, James P., Daniel T. Jones, e Daniel Roos. 1990. " *The machine that changed the world: based on the Massachusetts Institute of Technology 5-million dollar 5-year study on the future of the automobile"*. New York: Rawson Associates.

ANEXO A:Vantagens e Desvantagens de plataformas para gestão de desenvolvimentos e bugs.

Plataforma	Vantagens	Desvantagens
JIRA **(Sprint e Helpdesk)**	**(S e H)** Permite controlar as ações em tempo real **(S e H)** Ferramenta colaborativa. **(S e H)** Automatização de ações recorrentes (criação de tarefas) **(S e H)** Simples de gerir as tarefas atribuídas a cada membro **(S e H)** Simples para gerir/ criar um *roadmap* **(S e H)** Adequado ao uso de métodos *Agile* **(S)** Ferramenta bastante completa **(S e H)** Permite gerir as permissões de acesso e de edição. **(S e H)** Permite alocar as tarefas a períodos temporais **(S e H)** Permite controlar as ações em tempo real **(S e H)** Permite fazer *tracking* das tarefas de forma rápida **(S)** Cálculo automático de métricas adequadas para gestão de ***sprint,*** mas não para ***helpdesk***. **(S e H)** Permite integrações com outras plataformas **(S e H)** Permite anexar documentos junto de cada tarefa **(S e H)** Vem com *templates* e *workflows* e permite a personalização de novos e/ou alteração sobre estes. **(S e H)** Ferramenta escalável	**(S e H)** Elevada complexidade **(S e H)** Extração de informação no formato desejado difícil **(S e H)** Custo por utilizador **(H)** Não permite o cálculo de métricas personalizadas e adequadas
TRELLO **(Sprint e Helpdesk)**	**(S e H)** Permite controlar as ações em tempo real **(S e H)** Ferramenta colaborativa. **(S e H)** Baixa complexidade **(S)** Simples para gerir/ criar um roadmap **(S e H)** Intuitivo de usar **(S e H)** Rápida tomada de conhecimento do estado das tarefas **(S e H)** Permite gerir as permissões de acesso **(S e H)** Permite fazer *tracking* das tarefas **(S e H)** Permite integrações com outras aplicações **(S e H)** Permite anexar documentos junto de cada tarefa	**(S e H)** Difícil de gerir as tarefas de cada membro individualmente **(S e H)** Extração de informação difícil **(S e H)** Não permite gerir as permissões de edição **(S e H)** Não permite o cálculo de métricos – necessário o uso de outras ferramentas. **(S e H)** Custo por utilizador
Excel **(Sprint e Helpdesk)**	**(S e H)** Fácil de usar **(S e H)** Conhecido pela maioria dos utilizadores **(S e H)** Económico	**(S e H)** Ferramenta não colaborativa **(S e H)** Confuso pesquisar por informações e aceder ao histórico de informações **(S e H)** Não integra com outras ferramentas **(S e H)** Complexo de configurar

		(S e H) Não permite anexar ficheiros junto de outra informação **(S e H)** Não permitir gerir quem tem acesso à informação e quem a pode editar. **(S e H)** Necessário formatar todos os campos às necessidades, desde organização de informação ao cálculo de métricas
JIRA Service Desk **(Helpdesk)**	**(H)** Solução mais barata comparada a plataformas semelhantes (*BMC, Fresh service, Zendesk, Remedy*) **(H)** Integração direta com ferramenta de gestão de desenvolvimento (JIRA) **(H)** Rápido de colocar o serviço disponível para clientes **(H)** Cálculo de métricas automático, permitindo o estabelecimento de objetivos **(H)** Certificado pela *PinkVERIFY* **(H)** Vem com *templates* e *workflows* e permite a personalização de novos e/ou alteração sobre estes. **(H)** Ferramenta escalável **(H)** Fácil de acompanhar a evolução dos pedidos	**(H)** Custo por utilizador (apesar de ser bastante inferior quando comparado soluções da mesma natureza)

ANEXO B: Flowchart do funcionamento inicial do processo de *helpdesk*



ANEXO C: Inquérito de avaliação do serviço de helpdesk

Qualidade do serviço de helpdesk - até Fevereiro de 2018

Este questionário tem como objectivo analisar a forma como o helpdesk funcionava quando a figura do bombeiro era representada por um elemento da equipa tech. (o que aconteceu até fevereiro de 2018). NÃO devem ser dadas respostas tendo em conta o funcionamento do helpdesk actual (desde o fim do mês de fevereiro até ao momento). O preenchimento deste inquérito por questionário demora entre 3 a 5 minutos.

*Obrigatório

1. **Endereço de email** *

2. **Membro da equipa de:** *
Marcar apenas uma oval.

- ◯ Account Manager
- ◯ Equipa tech *Passe para a pergunta 11.*
- ◯ Operações
- ◯ Financeira & RH
- ◯ Marketing e Vendas
- ◯ Outra: _____

Nível de satisfação - até Fevereiro de 2018

Relembro que este questionário é em relação ao funcionamento de helpdesk antes da introdução da nova metodologia!

3. **Era utilizador do SPOKE?**
Marcar apenas uma oval.

- ◯ Sim
- ◯ Não *Pare de preencher este formulário.*

Nível de satisfação - até Fevereiro de 2018

Relembro que este questionário é em relação ao funcionamento de helpdesk antes da introdução da nova metodologia!

4. **Era comum deparar-se com erros/problemas ao executar as suas tarefas diárias?** *
Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

5. **Com que frequência se deparava com erros na execução das suas tarefas diárias?**
Marcar apenas uma oval por linha.

	Nunca	Raramente	Às vezes	Muitas vezes	Sempre
Linha 1	◯	◯	◯	◯	◯

6. **O tempo médio de espera até que o seu problema fosse resolvido era ...** * *Marcar apenas uma oval por linha.*

	Extremamente longo	Muito longo	Moderadamente longo	Pouco longo	Nada longo
Tempo de espera					

7. **Independentemente do que considerou ser o tempo médio de espera, o problema foi resolvido dentro de um período temporal que considera admissível ?** *
Marcar apenas uma oval por linha.

	Nunca	Raramente	Às vezes	Muitas vezes	Sempre
Tempo admissível					

8. **Após a resolução de cada problema era comum aparecer um semelhante?** * *Marcar apenas uma oval por linha.*

	Nunca	Raramente	Às vezes	Muitas vezes	Sempre
Reaparecimento do problema					

9. **Os problemas com que se costumava deparar eram ...**
* *Marcar apenas uma oval por linha.*

	Nunca	Raramente	Às vezes	Muitas vezes	Sempre
Mensagem de erro					
Erro tecnológico - bug					
O processo não funciona como eu esperava					
Não tinha conhecimento suficiente sobre o processo					
Sugestões de melhoria - ex.: poder filtrar por ordem alfabética					
Não existia a funcionalidade que eu precisava					

10. **Qual o nível de satisfação com o funcionamento do helpdesk até Fevereiro de 2018 (antes da mudança para a equipa de produto)** *
Marcar apenas uma oval por linha.

	Nada satisfeito	Pouco satisfeito	Moderadamente satisfeito	Muito satisfeito	Extremamente satisfeito
Nível de satisfação					

11. **Como avalia o funcionamento do helpdesk até Fevereiro de 2018 (antes da mudança para a equipa de produto)**
Marcar apenas uma oval por linha.

	Muito Mau	Mau	Razoável	Bom	Muito Bom
Linha 1					

Passe para a pergunta 19.

12. **Membro da equipa** *

Marcar apenas uma oval.

- ◯ BI/AI
- ◯ IT

13. **Costumavam ser-lhe atribuídos tickets de helpdesk?**

** Marcar apenas uma oval.*

- ◯ Sim
- ◯ Não

14. **Com que frequência?** *

Marcar apenas uma oval.

- ◯ Nunca
- ◯ Raramente (1 vez por sprint)
- ◯ Às vezes (+ do que uma vez por sprint)
- ◯ Muitas vezes (1 vez por dia)
- ◯ Sempre (+do que 1 vez por dia)

15. **Quanto tempo gastava, por sprint, com o helpdesk?** *Marcar apenas uma oval.*

- ◯ Nenhum tempo
- ◯ Pouco tempo
- ◯ Algum tempo
- ◯ Muito tempo
- ◯ Demasiado tempo

16. **Esse tempo gasto afectava a sua performance nas tarefas do sprint?** * *Marcar apenas uma oval.*

- ◯ Nunca
- ◯ Raramente
- ◯ Às vezes
- ◯ Muitas vezes
- ◯ Sempre

17. **Por favor, indique de que forma é que o tempo gasto afetava a sua performance (ex.: interrupção de tarefas, perturbação pelos stakeholders etc)** *

18. **Qual o nível de satisfação com o funcionamento do helpdesk até Fevereiro de 2018 (antes da mudança para a equipa de produto)** *
Marcar apenas uma oval por linha.

	Nada satisfeito	Pouco satisfeito	Moderadamente satisfeito	Muito satisfeito	Extremamente satisfeito
Nível de satisfação	◯	◯	◯	◯	◯

19. **Como avalia o funcionamento do helpdesk até Fevereiro de 2018 (antes da mudança para a equipa de produto)**
Marcar apenas uma oval por linha.

	Muito Mau	Mau	Razoável	Bom	Muito Bom
Linha 1	◯	◯	◯	◯	◯

Nível de satisfação da atual metodologia de helpdesk

Esta secção refere-se à metodologia que se implementou no final do mês de Fevereiro de 2018 (utilização do Trello sempre e uso de frameworks pré estabelecidas)

20. **Qual o nível de satisfação com o funcionamento do helpdesk atual (a partir do fim de Fevereiro)** *
Marcar apenas uma oval por linha.

	Nada satisfeito	Pouco satisfeito	Moderadamente satisfeito	Muito satisfeito	Extremamente satisfeito
Nível de satisfação	◯	◯	◯	◯	◯

21. **Como avalia o funcionamento do helpdesk atual (a partir do fim de Fevereiro)** *Marcar apenas uma oval por linha.*

	Muito Mau	Mau	Razoável	Bom	Muito Bom
Linha 1	◯	◯	◯	◯	◯

22. **Por favor, faça as sugestões/recomendações que considerar pertinentes para um melhor funcionamento do helpdesk** *

☐ Pretendo receber uma cópia das minhas respostas.

ANEXO D: Framework para especificação de requisitos

[Epic]| [User Story]

EPIC : [link to return to the 1st page of the dropbox with the specification of the epic and with the list of all user stories]

Job (When, I want to, So I can)

[Text with **When, I want to, So I can**)
Input: [text (What I give)]
Output: [text (What is given to me)]

1.1 Priority: [text (low, medium, high]

1.2 Size: [text]

1.3: Owner and release manager:

- Owner: [text]
- Release Manager: [text]

1.4 Business value: [text (low, medium, high)]

1.5 Users/Personas: [text (stakeholders - AMs, Operations, clients etc)]

1.6 Flow charts and sequence diagrams: [images]

1.7 Dependencies/precedencies on other User Stories:

[text - user stories that block or are blocked by this one]

- [Link]
- ...

1.8 Non-functional prototype: [link from figma - visual specification]

2. Tasks / Small jobs:

2.1 [Name of the task]:

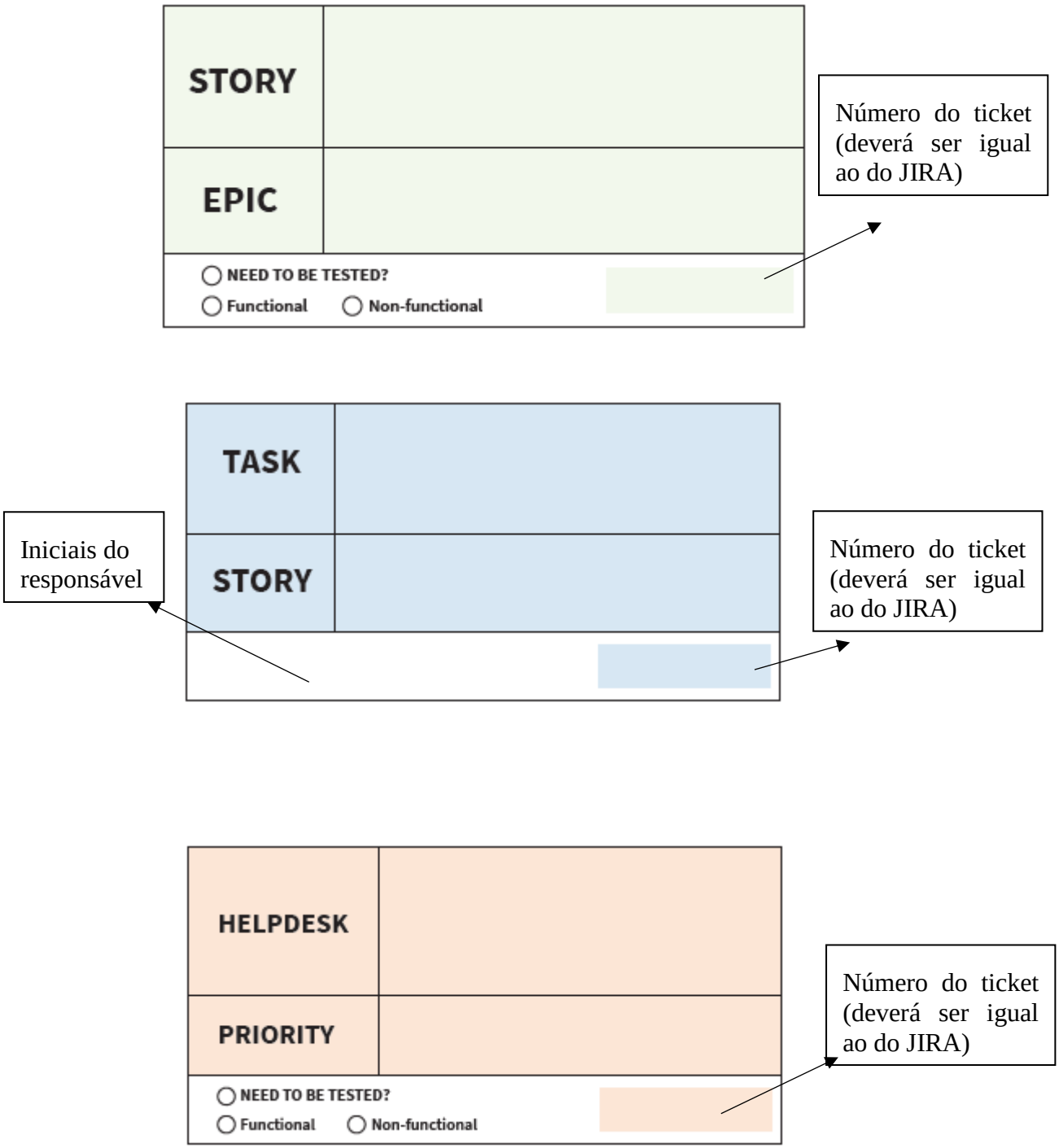
2.1.1 Definition
[text - brief description]
2.1.2 Acceptance criteria:
A. Functional:
[text]
B. Non-functional:
[text]

2.2 [Name of the task]:

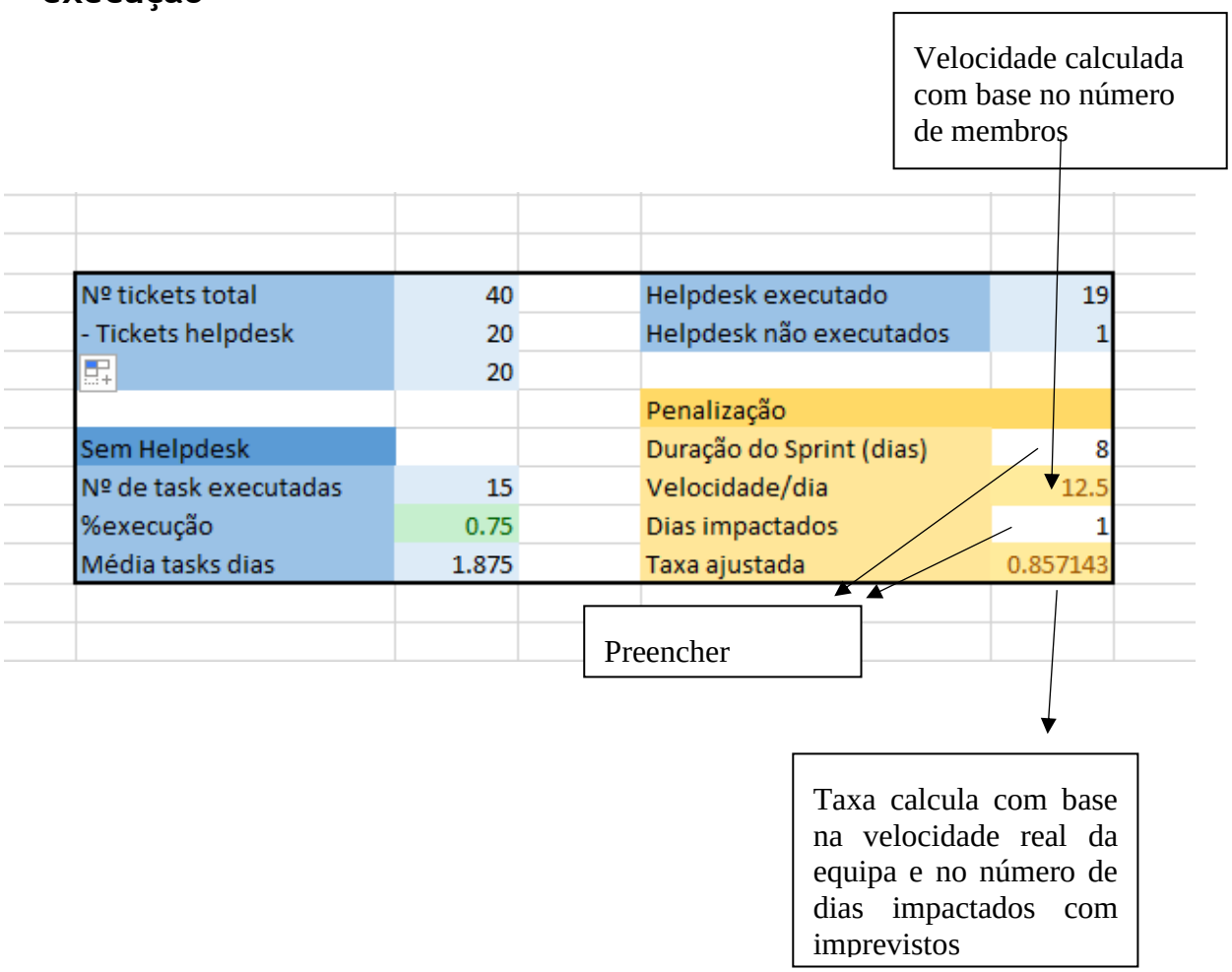
2.1.1 Definition
[text - brief description]
2.1.2 Acceptance criteria:
A. Functional:
[text]
B. Non-functional:
[text]

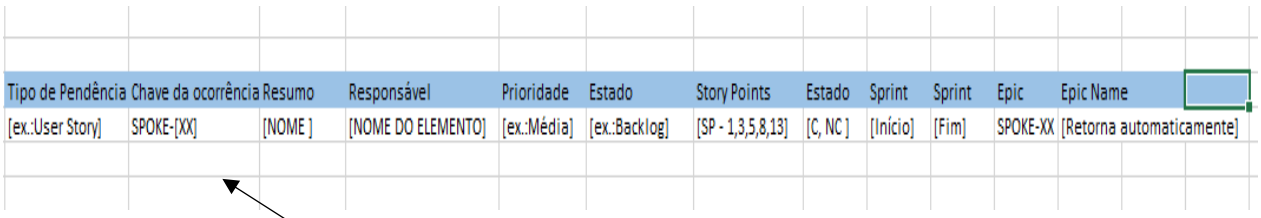
Add more in case of need.

ANEXO E: Framework dos cartões de gestão de *sprint*



ANEXO F: Estrutura da framework para cálculo das taxas de execução





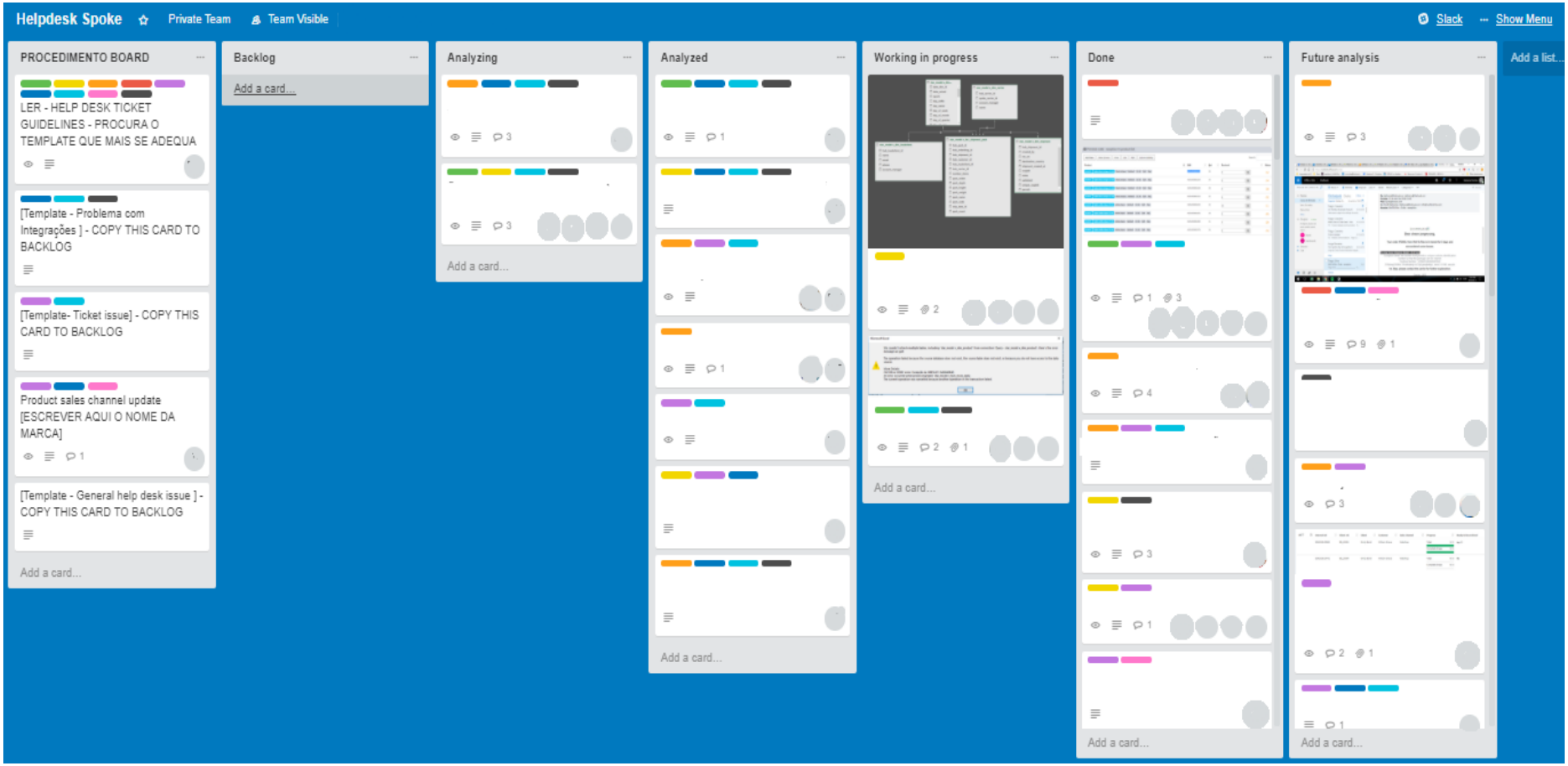
Tipo de Pendência	Chave da ocorrência	Resumo	Responsável	Prioridade	Estado	Story Points	Estado	Sprint	Sprint	Epic	Epic Name	
[ex.:User Story]	SPOKE-[XX]	[NOME]	[NOME DO ELEMENTO]	[ex.:Média]	[ex.:Backlog]	[SP - 1,3,5,8,13]	[C, NC]	[Início]	[Fim]	SPOKE-XX	[Retorna automaticamente]	

- Extrair a informação do JIRA, colar as informações nas colunas respetivas,com exepção da última.

- Preencher os campos:"Duração do Sprint", "Dias Impactados".

- A taxa é calculada de acordo com a informação dada, não contabilizando os pedidos de *helpdesk*, uma vez que este fogem da metodologia de *Scrum*.

ANEXO G: Estrutura do Trello para reportamento de problemas



ANEXO H: Framework para criação de Demos

Framework for Spoke Demo

← Back to the initial page

Esta página é acessível depois da seleção do processo ao qual o stakeholder pretende assistir.

Sendo a página anterior um conjunto de perguntas divididas por áreas.

[Name of the process]:

[Small explanation about the process and the importance of it. (What is the utility of the process? What is the propose ? Who can manage it?)]

There are [n] ways to do this process :

- [link and name] - [some instructions - ex.: What is the advantages of using this method]
  - [n] steps:
    - 1st [link and name] -
    - 2nd [link and name]
    - 3rd [link and name]
    - 4h [link and name]
- [link and name]
  - [n] steps
    - 1st [link and name] -
    - 2nd [link and name] -
    - 3rd [link and name] -
    - 4th [link and name] -

[Name]

1st step: [step name]
2nd step: [step name]
3rd step: [step name]
4th step: [step name]

A seguir a cada passo é introduzido um conjunto de texto e/ou bullet points seguido por vídeo ou imagens explicativas.

[Name]

1st step: [step name]
2nd step: [step name]
3rd step: [step name]
4th step: [step name]

ANEXO I: Framework para realização de testes para assegurar a qualidade das funcionalidades

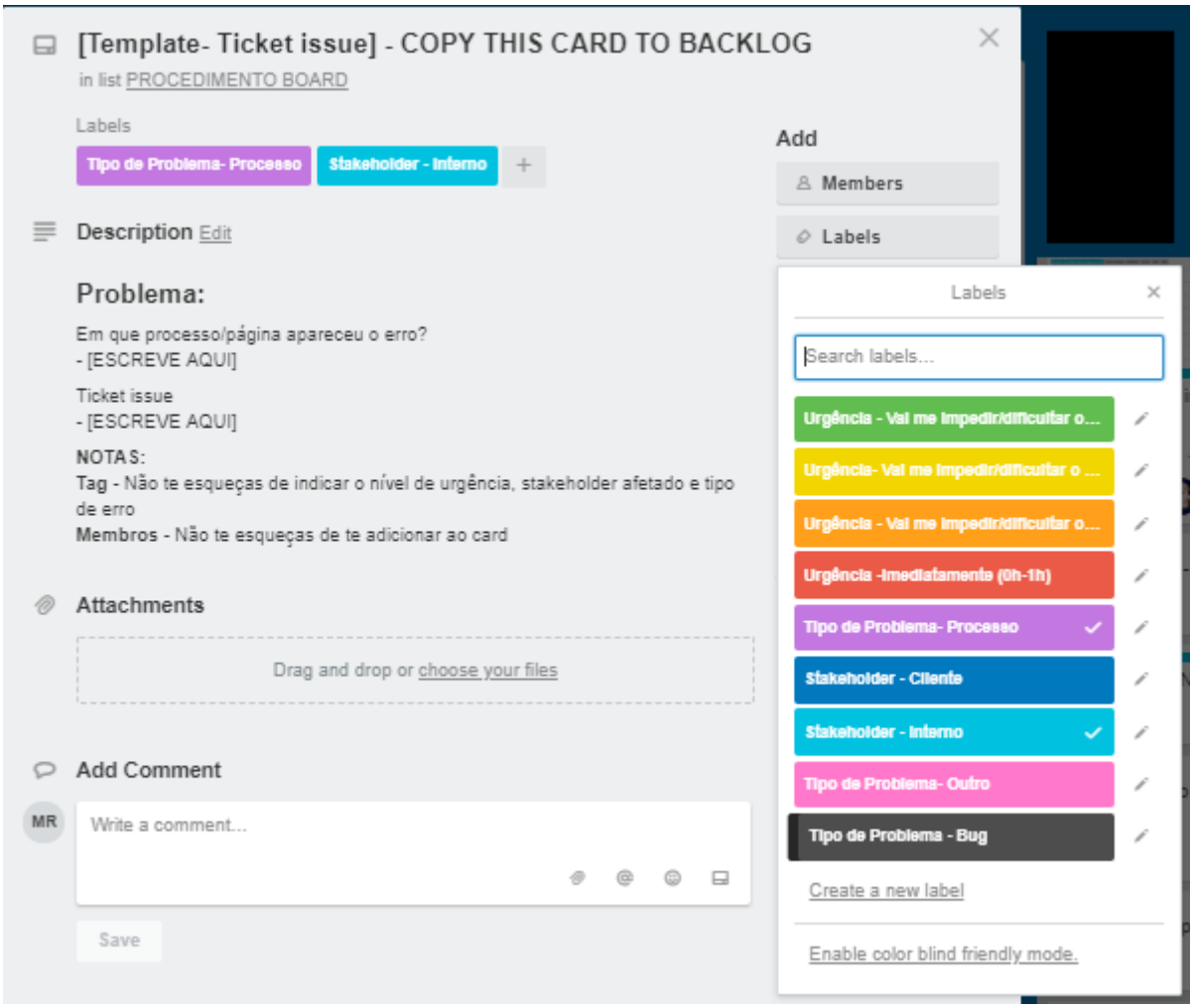
[Epic]| QA [User Story]

Por cada vez que a funcionalidade for submetida a teste deve criar-se uma coluna.

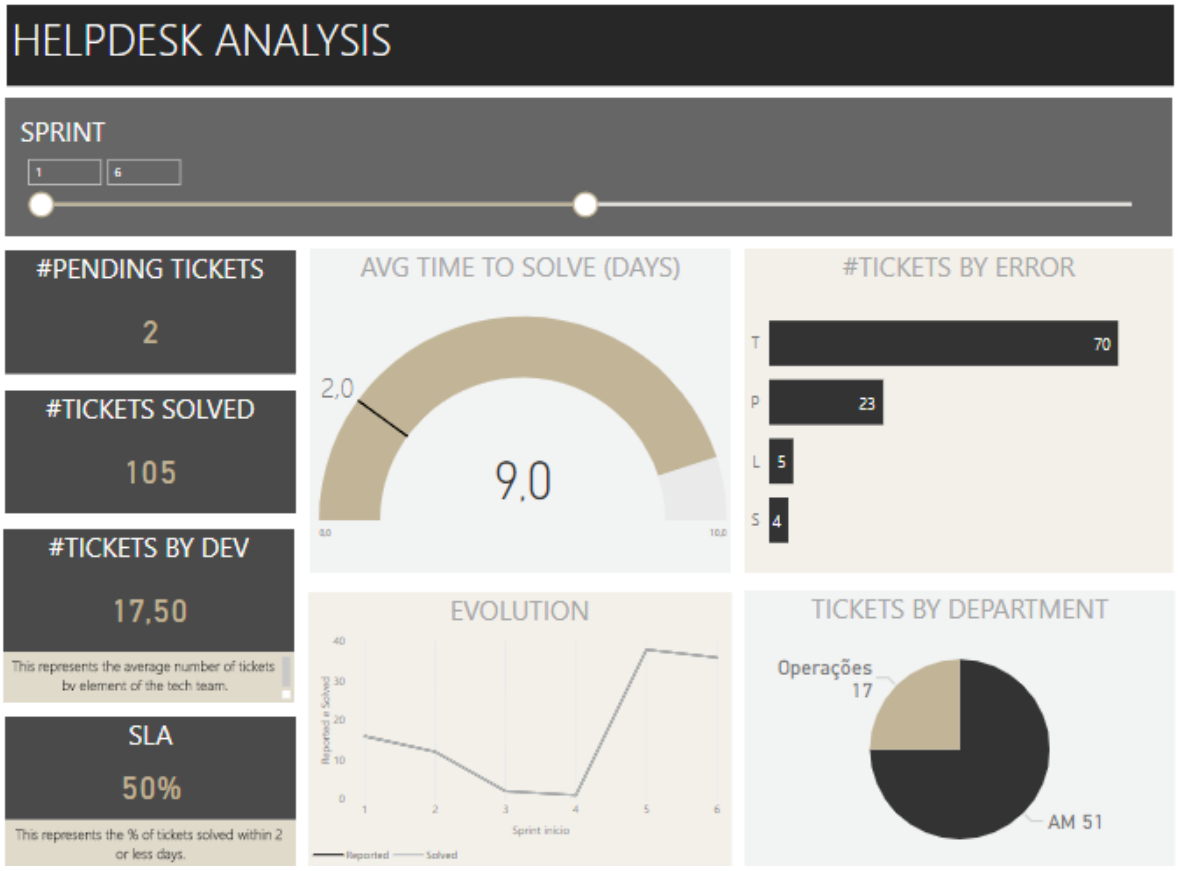
Element	When, I want to, So I can / Given, When, Then	1st test	2nd test
Epic - [text]			-
User Story 1 - [text]	when , I want , so I can	Role: [text] Sub-Feature: [text] Benefit : [text]	-
Acceptance Criteria 1 - [text]	Given [text] When [text] Then [text]	Inputs/Preconditions: [text] Actions/Triggers: [text] Outputs/Consequences: [text]	-
User Acceptance Test 1.1 (Non-functional) [text]	[1.Criteria]	[1. check or X]	[1. check or X]
User Acceptance Test 1.2 (functional) - [text]	[1.Criteria]	[1. check or X]	[1. check or X]
User Acceptance Test 1.3 (functional - [text]	[1.Criteria]	[1. check or X]	[1. check or X]

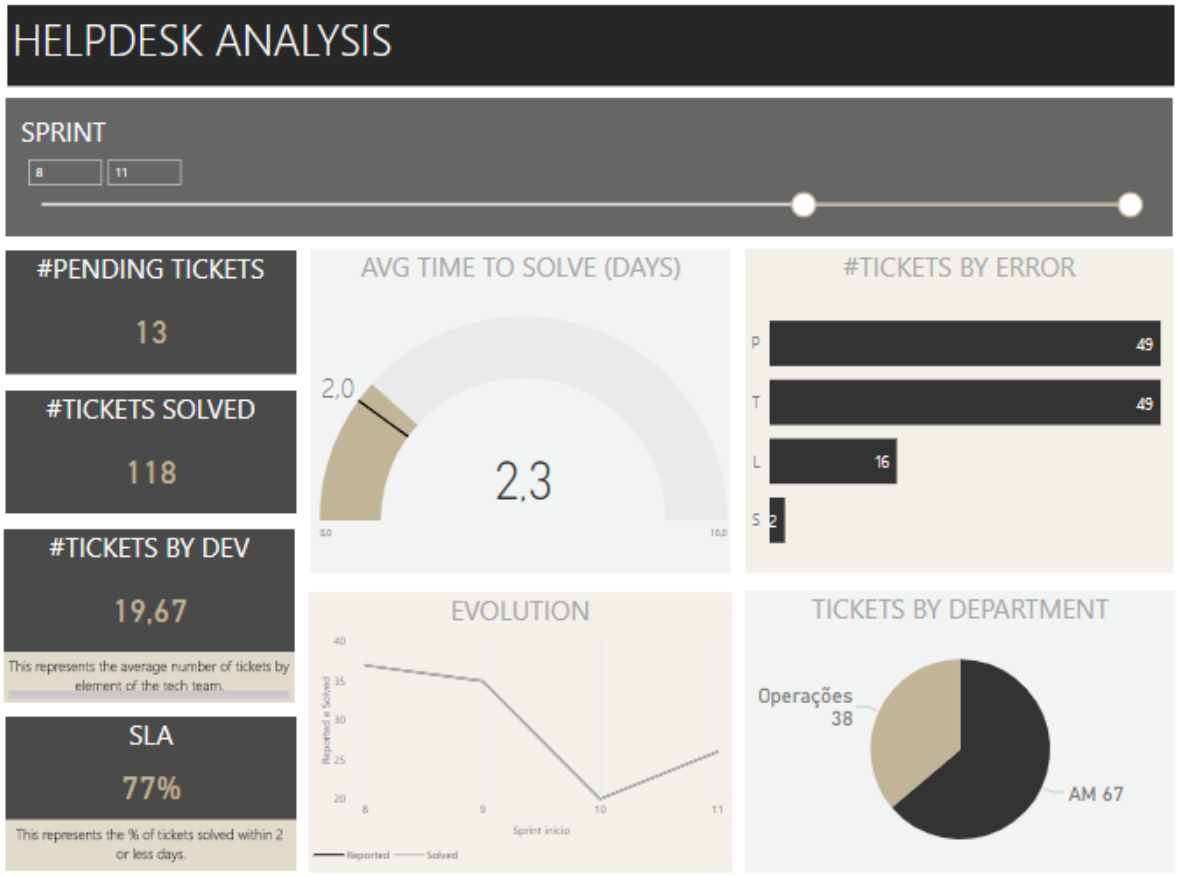
Devem criar-se tantas linhas quanto tipos de teste a serem realizados.

ANEXO J: Exemplo de uma framework para reportamento de um problema



ANEXO L: Gestão visual de métricas para helpdesk





Anexo M: Fluxo simples do novo sistema de *helpdesk*

A imagem que se segue foi utilizada na apresentação à empresa da nova versão do sistema de helpdesk, dai a simplicidade da mesma. As setas a verde representam processos de comunicação automáticos, sem intervenção humana, pela integração das várias plataformas : *Jira Service Desk*, *Slaasck*, *Slack* e plataforma da empresa. Assim, pretende-se com esta imagem transmitir a simplificação dos fluxos para o utilizador. Do lado interno esta simplicidade consegue-se através da realização de processos de definição complexos para montar e preparar o serviço, desde a criação *frameworks* para reportamento como a integração de plataformas. A segunda imagem é um *print* do novo ecrã a que o cliente tem acesso após ir à plataforma da empresa e solicitar assistência no *Slaask*.